ARA Software Engineering Curriculum Framework
Version 0.0

ARA Software Engineering Curriculum Framework

Table of Contents

I. Introduction 4
A. Background 4
B. Curriculum Framework 5
1. Software Engineering Competency Study, Phase II 5
2. Competency Measures 5

II. Software Engineering Competency Requirements 6
A. Acquisition 7
1. Competency Matrix 7
2. Elaboration of the Competency Requirements for the Acquisition Specialty 8
a. Basic Software engineering Knowledge 8
b. Computing Fundamentals 9
c. Software Product Engineering 9
d. Software Requirements 10
e. Software Management 11
f. Software Acquisition 11
g. Product Quality Control 12
h. Software Domains 12
B. Testing 13
1. Competency Matrix 13
2. Elaboration of the Competency Requirements for the Testing Specialty 14
a. Basic Software engineering Knowledge 14
b. Computing Fundamentals 14
c. Software Product Engineering 15
d. Software Testing 15
e. Software Management 16
f. Product Quality Control 16
g. Software Domains 16
C. Maintenance 17
1. Competency Matrix 17
2. Elaboration of the Competency Requirements for the Maintenance Specialty 18
a. Basic Software Engineering Knowledge 18
b. Computing Fundamentals 19
c. Software Product Engineering 19
d. Software Implementation 19
e. Software Maintenance Fundamentals 20
f. Software Management 20

III. Software Engineering Curriculum Specification 21
A. Curriculum Organization 21
B. Learning Module Specifications 22
1. Basic Software Engineering Knowledge 22
2. Computing Fundamentals 24
3. Software Domains 25
4. Software Management (Fundamental) 26
5. Software Management (Advanced) 27
6. Product Quality Control 29
7. Software Product Engineering (Basic) 31
8. Software Product Engineering (Intermediate) 32
9. Software Product Engineering (Advanced) 34
10. Software Implementation 36

11. Software Requirements 38
12. Software Testing 39
13. Software Maintenance 41
14. Software Acquisition 42

IV. References 44

Appendix A: Learning Resources 45

ARA Software Engineering Curriculum Framework

I. INTRODUCTION

A. BACKGROUND

Software is playing an increasingly important and central role in all aspects of daily life: in government, banking and finance, education, transportation, entertainment, medicine, agriculture, and law. The number, size, and application domains of programs being developed has grown dramatically; as a result, billions are being spent on software development, and the livelihood and lives of millions directly depend on the effectiveness of this development. Unfortunately, there are severe problems in the cost, timeliness, and quality of many software products; and even more serious, is the affect that quality problems can have on the safety-critical elements of software that is central to many aviation systems.

The FAA, the air traffic control systems, individual aircraft and pilots, and others who plan for and use the National Air Space, are relying more on software for their critical functions. Given this all-pervasive nature of software in the FAA's environment, the need for software engineers and software-knowledgeable personnel is growing rapidly in all areas and levels of the FAA. The need to develop a framework for determining, assessing, and improving software engineering competencies within the FAA has becoming increasingly urgent.

In 1998, faculty in the Computer Science Department at Embry-Riddle Aeronautical University worked on a FAA funded project titled "Study of FAA-ARA Software Engineering Competencies" (referred to in this report as "Phase I"). This project was designed to help improve the software engineering knowledge and abilities of FAA-ARA (Associate Administrator for Acquisitions and Research) personnel. It sought to identify and categorize the body of knowledge for the discipline of software engineering, and correlate this body of knowledge with the ICIP (Intellectual Capital Investment Plan) documented responsibilities, activities, and competencies in ARA software-related roles.

In support of the project objectives, the following deliverables were produced:

- *Description of Software Engineering Knowledge* - a structured description of the software engineering body of knowledge (SwE-BOK)
- *A Software Engineering Competency Model* – a framework to establish the relationship between ICIP defined ARA roles, the SwE-BOK and FAA-iCMM process areas.

Both these deliverables were published in *FAA-ARA Software Engineering Competency Study: Final Report* [1] and the SwE-BOK was published as a Technical Report of the Software Engineering Institute [2].

The project work highlighted the pervasive nature of software in the activities and responsibilities of the ARA roles. One of the consequences of Phase I of the project was recognition of the problem of defining ICIP roles with a single model. A particular ICIP role, when viewed in the context of the actual jobs associated with the role, might encompass a rather complex subdivision and hierarchy of sub-roles. For example in its study of the Software Engineering role the project team discovered that although there were technical competencies common to all personnel serving in software engineering roles, there were at least three specialty areas (acquisitions, testing, and maintenance) that had additional distinct competency requirements.

B. CURRICULUM FRAMEWORK

1. Software Engineering Competency Study, Phase II

In 1999, a continuation of the FAA-ARA software engineering competency study begun in 1998. The chief objective of this part of the study, called "Phase II", is to improve the software engineering knowledge and abilities of FAA software engineers by

- updating and refining the ARA Software Engineering Competency Model [1] for the software engineering role, with three specialty areas: acquisition, maintenance, and testing; and
- developing a description and outline for an ARA software engineering curriculum.

This document contains information, guidance, and a structure for designing and implementing a curriculum for assessing and improving the competencies of ARA software engineers. The "curriculum framework" described herein includes a detailed description of the software competencies required in each specialty area. In addition, we outline and specify a set of curriculum modules that support acquiring the knowledge embodied in the competency requirements.

2. Competency Measures

As part of this work, a hierarchical system of evaluating and assessing the “depth of software engineering knowledge “ (based on the SwE-BOK) of activities associated with the software engineering role has been formulated. These depth of knowledge measures (with the exception of the Mastery level) are based on earlier work [1] and are defined as follows:

A – Awareness: Represents a level of knowledge about a software engineering subject so that an individual

- has awareness of the existence and the context of the subject within the subject
- can provide a general, informal explanation about the subject
- can identify references (human/ literature) that provide greater depth of knowledge about the subject

U- Understanding: Represents a level of knowledge about a software engineering subject so that an individual

- can explain the subject through definition and example, and appreciates the effort needed to perform work related to the subject
- can monitor the progress of the work related to the subject
- can evaluate the quality of the work related to the subject

E – Execution: Represents a level of knowledge about a software engineering subject so that an individual

- can apply the knowledge to produce software engineering products
- can apply the knowledge to analyze and evaluate methods and techniques in the subject
- can inform others about the content and practices associated with the subject

M -Mastery: Represents a level of knowledge about a software engineering subject so that an individual

- can educate and mentor others about the content and practices associated with the subject
- can provide consultation and expert advice about the subject
- can describe the connections and interactions between the subject and other knowledge components, and provide judgement as to the competency required for tasks associated with the subject

In Section II we use the competency measures to designate the level of knowledge and practice needed by software engineers to carry out their tasks. We also recommend (in Section III) minimum training necessary to attain this knowledge. In all cases the training must be supplemented by on-the-job experience in order to reach the desired competency level; this is especially true for knowledge components with designations at the "execution" level.

It should be noted that, in Section II, none of the specialties have knowledge designated at the "mastery" level. Since we are identifying minimum competency requirements for each of the specialties, the first three levels of competency are sufficient measures. However, it would be expected that in the FAA there would be individual engineers that possess mastery level competency for one or more of the knowledge components (e.g., a maintenance software engineer with mastery level competency for the Software Maintenance knowledge component).

II. SOFTWARE ENGINEERING COMPETENCY REQUIREMENTS

In this section we provide a description of the minimum competency requirements for three specialty areas for the ARA Software engineering role: acquisition, testing, and maintenance. The scope of the description is confined to required knowledge about software engineering. Competency in other areas such as system engineering, or human factors, or in a specific application domain (e.g. ATC) or product line is not addressed in this document. The description of each specialty consists of a competency matrix along with a narrative elaboration of each matrix element. Each specialty matrix describes the minimum knowledge required in the specialty, in terms of a set of knowledge components, and the level of competency necessary for each knowledge component.

In Phase I of this project a software engineering body of knowledge (SwE-BOK) [2] was developed using a classification scheme based on a hierarchical decomposition consisting of Knowledge Categories, Knowledge Areas, and Knowledge Units. The SwE-BOK classifies software engineering knowledge into four knowledge categories:

1. Computing Fundamentals
2. Software Product Engineering
3. Software Management
4. Software Domains

Each knowledge category is subdivided into a number of knowledge areas. For example, Software Product Engineering is divided into five knowledge areas:

2.1 Software Requirements Engineering
2.2 Software Design
2.3 Software Coding
2.4 Software Testing
2.5 Software Operation and Maintenance.

Each Knowledge area is further subdivided into a number of knowledge units. For example, the Software Requirements Engineering is divided into three knowledge units:

2.1.1 Requirements Elicitation
2.1.2 Requirements Analysis
2.1.3 Requirements Specification

The competency model for a specific software engineering specialty is represented by a matrix that is designed in the following fashion:

- The first column of the competency model names the knowledge components for the specialty area. The knowledge component is represented by a collection of individual knowledge elements (categories, areas, and units) from the SwE-BOK.
- The next four columns in the matrix represent knowledge levels (Awareness, Understanding, and Execution) that are used in this model. Each labeled element of the matrix represents a knowledge component and the specific depth of the knowledge, as a minimum, that the software engineer must possess in order to perform his/her tasks. The below matrix represents a partial representation of the competency model for a software engineer with testing responsibilities. In this matrix, the "Basic Software Engineering Knowledge" component includes the knowledge categories 1, 2, 3, and 4; and it requires competency at the "awareness" level. However, the "Product Quality Control" knowledge component requires competency at the execution level for the knowledge units 2.5.1, 3.3.1, 3.3.2, and 3.3.3. What we specify in this matrix is that, as a minimum, the software tester needs to possess an awareness level of "Basic Software Engineering Knowledge" and requires an execution level knowledge of "Product Quality Control". Notice that the lower level competency requirements for the "Basic" component (awareness of 1, 2, 3, 4) are augmented by "deeper" competency requirements in some of the units for the "Product Quality Control" component (execution of 2.5.1, 3.3.1, 3.3.2, 3.3.3).

Knowledge Component	Awareness	Understanding	Execution
Basic Software Engineering Knowledge 1 Computing Fundamentals 2 Software Product Engineering 3 Software Management 4 Software Domains	a		
Product Quality Control 2.5.1 Software Installation and Operation 3.3.1 Software Quality Assurance 3.3.2 Verification and Validation 3.3.3 Software Metrics			f

Finally, the designator under the awareness level (e.g., a), corresponds to a narrative that provides additional information and rationale for the competency requirement.

A. ACQUISITION

1. Competency Matrix

Specialty Area: Acquisition

Specialty Description: The acquisition software engineer serves an interface between the FAA (users/customers that need to acquire a software system), the development organization (a contractor that will be chosen to develop the software system), and the system engineer responsible for overall definition, development, integration and deployment of the acquired software system. Therefore, the acquisition software engineer has a wide range of activities, which extend from eliciting requirements, through monitoring software development and acceptance testing, to support the transition to the deployment and maintenance phase.

Knowledge Component	Awareness	Understanding	Execution
Basic Software Engineering Knowledge 1 Computing Fundamentals 2 Software Product Engineering 3 Software Management 4 Software Domains	a		
Computing Fundamentals 1.1.1 Basic Data Structures 1.2.2 Computer System Organization 1.2.3 Alternative Architectures 1.2.4 Communications and Networks 1.4 Operating Systems		b	
Software Product Engineering 2.1 Software Requirements Engineering 2.2.1 Architectural Design 2.2.2 Abstract Specification 2.2.3 Interface Design 2.2.4 Data Structure Design 2.4.2 Integration Testing 2.4.3 System Testing 2.4.4 Performance Testing 2.4.6 Installation Testing 2.4.7 Test Documentation 2.5.1 Software Installation and Operation 2.5.2 Software Maintenance Operations 2.5.5 Software Reengineering		c	
Software Requirements 2.1.1 Requirements Elicitation			d
Software Management 3.1 Software Project Management 3.2 Software Risk Management 3.4 Software Configuration Management 3.5 Software Process Management			e
Software Acquisition 3.6 Software Acquisition			f
Product Quality Control 2.4.5 Acceptance Testing 3.3 Software Quality Management			g
Software Domains 4.3 Human-Computer Interaction 4.6 Real-Time Systems		h	

2. Elaboration of the Competency Requirements for the Acquisition Specialty

a. Basic Software engineering Knowledge

Description This component includes basic knowledge about software engineering as a discipline. It covers basic knowledge about the fundamentals of computing, software product development, software management, and various software application domains. It embodies minimum prerequisite knowledge for more detailed and specialized study of software engineering.

Competency Requirements Engineers must have awareness of the existence and the context of the basic elements of software engineering. They must be able to provide a general, informal explanation of the terminology, concepts, and techniques associated with the discipline of computing; and they must be able to identify references that provide greater depth of knowledge about software engineering.

Rationale There are a wide variety of activities and tasks engaged in by ARA software engineers that have responsibility for acquisition of software. The knowledge, skills, and capabilities required depend on the size and complexity of the software and the specific contractor requirements. The chief responsibility for an "acquisition" software engineer is to monitor the software development through the initial phases of development: requirements specification, design, implementation, and system and acceptance testing; and to support transition of the software to the deployment and maintenance phases. The engineer must have familiarity with an extensive array of basic computing and software engineering knowledge, terms and concepts across all four knowledge categories specified in [2]: Computing Fundamentals, Software Product Engineering, Software Management, and Software Domains. In-depth knowledge of each area and unit is not required, but general awareness of the content and significance of each unit is important.

b. Computing Fundamentals

Description This component covers knowledge, concepts, and principles of computing that are essential to the development of real-time embedded and distributed computing systems. It includes knowledge about computer system organization and operation, communication and network essentials, various computer architectures, and the fundamentals of operating systems.

Competency Requirements Engineers must be able to explain computer system terminology and concepts through definition and example. They must be able to monitor and evaluate the development of software that interacts with and controls external devices.

Rationale In order for acquisition software engineers to be able to interact effectively with contracted software developers (analysts, designers, programmers, and quality engineers) it is necessary for the engineers to have an understanding of fundamental computing terms and concepts. In order to make acquisition evaluations and decisions it is important that these engineers possess basic understanding of the organization and operation of digital computers systems, and communication and network systems.

c. Software Product Engineering

Description This component covers fundamental knowledge about the engineering of a software product: software requirement engineering, software design, software testing and software maintenance.

Competency Engineers must be able to explain, through definition and example,

Requirements software product engineering terminology and concepts. They should understand the effort required for the various software product engineering tasks. They must be able to monitor and evaluate the development of a software product throughout its development life-cycle.

Rationale Since the primary responsibility of an acquisition software engineer is to monitor and support the development of a software product it is important for such engineers to have a good comprehension of the activities associated with software product development. In addition, they should have sufficient knowledge to effectively judge and help avoid maintainability and evolution problems. This requires a sound understanding of requirements engineering, design specification (including software architecture, interface and data design), and software testing beyond the unit level. Also, in the beginning of a project the acquisition engineer must interact with system engineering in order to insure that the proposed software development activities are achievable. Since acquisition software engineers have to make judgements about the viability and effectiveness of the contractor development methods and processes, they need to be able to read, comprehend, and evaluate the quality of requirements specifications, high-level design documents, and system test plans. To insure good quality, they must also be able to trace requirements through all phases of the software development life cycle

d. Software Requirements

Description This component covers knowledge that supports the systematic development of a complete understanding of the problem domain. It includes knowledge about methods and techniques for uncovering, discovering and communicating functional and non-functional requirements and constraints; it provides a foundation for decomposing a problem into intellectually manageable pieces by using objects, functions and states.

Competency Requirements Engineers must be able to elicit and determine software needs for a system. They must be able to support system developers and users in problem definition and software requirements analysis and specification. They must be able to analyze and evaluate software collection methods and techniques. They must be able to inform others about the content and practices of requirement elicitation.

Rationale The acquisition software engineer provides a critical interface been the FAA as a user/customer and the contractor as the developer; he/she must support the effort to establish a common understanding of the requirements to be addressed by the software product. Hence, it is important for the engineer to have knowledge and experience with methods and techniques for uncovering, discovering, and communicating functional and non-functional requirements and constraints. In addition, the acquisition engineer should have knowledge about the modeling of software requirements in the information, functional, data, and behavioral domains of a problem. An acquisition engineer must be equipped to make a trade-off analysis between functional requirements and the constraints on a system, along with all the derived requirements of a system, which

highlight the affect on development cost and schedule.

e. Software Management

Description The component provides comprehensive coverage of the concepts, methods and techniques for managing the development of software products. It includes knowledge about software activities concerned with project management, management of risk, the configuration of a software system, knowledge about how to produce high-quality software and the development and improvement of software processes.

Competency Requirements Engineers must be able to analyze and evaluate product management elements: project plans, configuration management systems, risk plans, quality assurance activities, and software development processes. They must be able to interpret and analyze software metrics, and assess process and product quality. They must be able to analyze and evaluate various software management methods, and to inform others about the content and practices of software management.

Rationale The primary role of acquisition software engineers is to monitor the development/acquisition of a software system. For many FAA systems the size, complexity, and criticality of the software components require significant and long-term management oversight by the acquisition engineer. They must be able to review a software development plan, evaluate a project budget, monitor project progress, participate in a software review, and study and analyze software measurement data. This requires knowledge about a wide range of software management activities: project planning, quality assurance, software metrics, configuration management, risk management, and software processes.

f. Software Acquisition

Description This component provides comprehensive coverage of issues associated with software acquisition. It includes knowledge about the concepts, methods, processes, procedures, and techniques associated with procurement, contracting, performance evaluation, software management, and software quality control.

Competency Requirements Engineers must be able to carry out the software acquisition activities concerned with procurement, contracting, and performance evaluation. They must be able to organize and plan software acquisition activities, track product development, and assess the need for future support of a system. They must be able to analyze and evaluate various acquisition methods, and to inform others about the content and practices of software acquisition.

Rationale A software engineer working in the acquisition area must have explicit knowledge and experience about acquiring a custom software system from software developers that are independent of the FAA. This includes knowledge about acquisition activities such as procurement, contracting, performance evaluation, and providing for future support of the software system. Knowledge about the appraisal and acquisition of COTS software, and its use and

integration in software systems is becoming increasing critical for acquisition engineers.

g. Product Quality Control

Description This includes in-depth knowledge about practices that are necessary for producing high quality software. It covers material on quality assurance, formal review, software metrics, and assessment and analysis of software quality.

Competency Requirements Engineers must be able to apply knowledge about software quality to participate in product reviews, assess the quality of a software product, and evaluate an organization's software assurance capability. They must be able to analyze and evaluate various software quality assurance methods, and to inform others about the content and practices associated with the software quality control.

Rationale A key responsibility of an acquisition software engineer is to help assure that acquired software satisfies its requirements and is defect free. The acquisition engineer may participate in design reviews, review quality plans, examine the results of testing, and verify requirements tracing throughout software development. This requires knowledge and experience with the concepts, methods, and activities necessary to confirm that the software requirements are carried through each phase in the software life cycle. The engineer must have execution level knowledge about requirements tracing, walkthroughs, inspections, and validation techniques.

h. Software Domains

Description The component consists of two parts: one on human computer interaction and one on real-time systems. The human computer interaction part covers user interfaces, computer graphics, and hypertext/hypermedia. The real-time systems part includes knowledge about basic properties of real-time application software and the development of real-time software systems.

Competency Requirements Engineers must be able to explain the terminology and concepts, through definition and example, in the software domains of human computer interaction and real-time systems. They must be able to monitor and evaluate the development of software that involves significant requirements and functionality related to human computer interaction and real-time computing.

Rationale Because of the nature of FAA computing requirements, intensive user interaction and real-time embedded computing are typical characteristics of its software systems. Hence, it is important that acquisition software engineers have familiarity and understanding in other "specialty" software domains; in particular the engineer should have knowledge of human computer interaction and real-time systems.

B. TESTING

1. Competency Matrix

Specialty Area: Testing

Specialty Description: The test software engineer is responsible for validating, through software testing, a software system's required functionality, its conformance to FAA's standards and procedures, and its agreement with pre-defined development, design and operational environment constraints. The test engineer plans, develops, implements, and analyzes, and documents the testing of a software system.

Knowledge Component	Awareness	Understanding	Execution
Basic Software Engineering Knowledge 1 Computing Fundamentals 2 Software Product Engineering 3 Software Management 4 Software Domains	a		
Computing Fundamentals 1.1.1 Basic Data Structures 1.2.2 Computer System Organization 1.2.3 Alternative Architectures 1.2.4 Communications and Networks 1.3.2 Discrete Mathematical Structures 1.3.5 Probability and Statistics 1.4 Operating Systems 1.5.2 Programming Paradigms		b	
Software Product Engineering 2.1 Software Requirements Engineering 2.2.1 Architectural Design 2.2.2 Abstract Specification 2.2.3 Interface Design 2.2.4 Data Structure Design 2.3.1 Code Implementation 2.4.1 Unit Testing		c	
Software Testing 2.4.2 Integration Testing 2.4.3 System Testing 2.4.4 Performance Testing 2.4.5 Acceptance Testing 2.4.6 Installation Testing 2.4.7 Test Documentation			d
Software Management 3.1 Software Project Management 3.2 Software Risk Management 3.3 Software Quality Management 3.4 Software Configuration Management 3.5 Software Process Management		e	
Product Quality Control 2.5.1 Software Installation and Operation 3.3.1 Software Quality Assurance 3.3.2 Verification and Validation 3.3.3 Software Metrics			f
Software Domains		g	

4.3 Human-Computer Interaction 4.6 Real-Time Systems			

2. Elaboration of the Competency Requirements for the Testing Specialty

a. Basic Software engineering Knowledge

Description This component includes basic knowledge about software engineering as a discipline. It covers basic knowledge about the fundamentals of computing, software product development, software management, and various software application domains. It embodies minimum prerequisite knowledge for more detailed and specialized study of software engineering.

Competency Requirements Engineers must have awareness of the existence and the context of the basic elements of software engineering. They must be able to provide a general, informal explanation of the terminology, concepts, and techniques associated with the discipline of computing; and they must be able to identify references that provide greater depth of knowledge about software engineering.

Rationale There are a wide variety of activities and tasks engaged in by software engineers with responsibilities for software testing. The knowledge, skills, and capabilities required depend on the size and complexity of the software and the type of testing (functional testing, operational testing) being performed. The chief responsibility of a "testing" software engineer is to verify and validate the quality of the software based on the software requirements (both functional and non-functional). Examples of software testing responsibilities include the verification of a system's functionality, conformance to a set of FAA's standard operating procedures, conformance to pre-defined design constraints, environmental constraints and standards, and conformance to other sets of constraints and standards. As a result, the engineer must have familiarity with an extensive array of basic computing and software engineering knowledge, terms and concepts across all four knowledge categories specified in [2]: Computing Fundamentals, Software Product Engineering, Software Management, and Software Domains. In-depth knowledge of each area and unit is not required, but general awareness of the content and significance of each unit is important.

b. Computing Fundamentals

Description This component covers knowledge, concepts, and principles of computing that are essential to the development of real-time embedded and distributed computing systems. It includes knowledge about computer system organization and operation, communication and network essentials, various computer architectures, and the fundamentals of operating systems.

Competency Requirements Engineers must be able to explain, through definition and example, terminology and concepts associated with the fundamentals of computing. They can use the elements of computing and statistics to monitor and evaluate effective software testing.

Rationale It is important that test engineers possess basic understanding of discrete mathematical structures, data structures, computer system organization and operation, communication and network essentials, various computer architectures, and the fundamentals of operating systems. In addition, since effective software testing requires attention to quantitative analysis and management of defects, test engineers must have an understanding of statistical techniques.

c. Software Product Engineering

Description This component provides overview information about software requirements engineering and software design. Issues such as requirement elicitation, analysis, specification, and tracking are covered. Also, Issues associated with software design such as architectural, interface, and data structure design are addressed. In addition, this component includes knowledge that supports the implementation and testing of a small software module.

Competency Requirements Engineers must be able to explain, through definition and example, software product engineering terminology and concepts. They should understand the effort required for the various software product engineering tasks. They must be able to describe the tasks necessary for the implementation and testing of a small software module.

Rationale Since the primary responsibility of a software engineer with testing responsibility is to monitor and verify the quality of a software product it is important for such engineers to have a good comprehension of the activities associated with software product development. In particular, the engineer must have a sound understanding of requirements engineering, design specification (including software architectures, interface design, and data design), software coding (on occasion the test engineer may have to prepare test drivers and other elements of the test environment), and a good understanding of unit testing. The engineer may work with a contractor as part of a development team, monitoring their testing activity. As such they must be able to read, comprehend, and evaluate the quality of requirement specifications and high-level design documents; and understand the role of requirements tracing to ensure all requirements have been properly implemented.

d. Software Testing

Description This component covers in-depth knowledge and activities that are associated with software testing. It includes material on the software testing life cycle, testware development, software testing techniques and methods, and software testing metrics.

Competency Requirements Engineers must be able to carry out the software testing activities concerned with validating that a software product satisfies its requirements. They must be able to apply knowledge to analyze and evaluate the effectiveness of software testing. They must be able to inform others about the content and practices of software testing.

Rationale Test engineers need both deep and broad knowledge about software testing; they are involved in a multi-stage process that

consists of activities for validating the software product, from the most primitive elements up to a fully integrated system. Therefore, test engineers must have execution knowledge of test documentation, and integration, system, performance, acceptance, and installation testing.

e. Software Management

Description The component provides overview knowledge about software management responsibilities, methods, and activities. It includes basic knowledge about the terminology and techniques associated with project planning, quality assurance, software metrics, configuration management, risk management, and software processes.

Competency Requirements Engineers must be able to explain, through definition and example, software management terminology and concepts. They should be able to identify and evaluate management tasks associated with software testing.

Rationale The test software engineer must possess the appropriate knowledge to be able to deliver a tested product on time and within budget. Hence, such engineers must have a good understanding of the following: project planning, quality assurance, software metrics, configuration management, risk management, and software processes.

f. Product Quality Control

Description This includes in-depth knowledge about practices that are necessary for producing high quality software. It covers material on quality assurance, formal review, software metrics, and assessment and analysis of software quality.

Competency Requirements Engineers must be able to apply knowledge about quality planning and control, product and process metrics, and verification and validation to assess and improve the quality of a software product. They must be able to inform others about the content and practices associated with software quality control.

Rationale The primary responsibility of the test software engineer is to validate the quality of the software product. This means it is important that he/she be familiar with concepts, methods, techniques, procedures, and standards for producing high-quality software products. It is beneficial that the engineer have execution knowledge about quality planning and control, verification and validation activities, measurement of product and process attributes, and assessing effective software operation.

g. Software Domains

Description The component consists of two parts: one on human computer interaction and one on real-time systems. The human computer interaction part covers user interfaces, computer graphics, and hypertext/hypermedia. The real-time systems part includes knowledge about basic properties of real-time application software

and the development of real-time software systems.

Competency Requirements Engineers must be able to explain the terminology and concepts, through definition and example, in the software domains of human computer interaction and real-time systems. They must be able to monitor and evaluate the development of software that involves significant requirements and functionality related to human computer interaction and real-time computing.

Rationale Because of the nature of FAA computing requirements, intensive user interaction and real-time embedded computing are typical characteristics of its software systems. Hence, it is important that test software engineers have familiarity and understanding in other "specialty" software domains; in particular the engineer should have knowledge and understanding of human computer interaction and real-time systems.

C. MAINTENANCE

1. Competency Matrix

Specialty Area: Maintenance

Specialty Description: The maintenance software engineer is responsible for making changes to existing FAA software systems. This includes changes that correct software defects, enhance the functionality of the software, or change the operating environment for the software. Specifically, maintenance engineers receive and analyze a maintenance request, identify alternative solutions, chose the best solution, and then design, implement, document, and test the solution.

Knowledge Component	Awareness	Understanding	Execution
Basic Software Engineering Knowledge 1 Computing Fundamentals 2 Software Product Engineering 3 Software Management 4 Software Domains	a		
Computing Fundamentals 1 Computing Fundamentals		b	
Software Product Engineering 2.1 Software Requirements Engineering, 2.2 Software Design, 2.4 Software Testing, 2.5 Software Operation and Maintenance)			c
Software Implementation 1.1 Algorithms and Data Structures 1.5.2 Programming Paradigms 1.5.3 Programming Language Design and Implementation 2.2.5 Algorithm Design 2.3 Software Coding 2.4 Software Testing 4.2 Database Systems			d

4.3 Human-Computer Interaction 4.6 Real-Time Systems			
Software Maintenance Fundamentals 2.5.1 Software Installation and Operation 2.5.2 Software Maintenance Operations 2.5.3 Software Maintenance Process			e
Software Management 3.1 Software Project Management, 3.2 Software Risk Management 3.3 Software Quality Management 3.4 Software Configuration Management 3.5 Software Process Management		f	

2. Elaboration of the Competency Requirements for the Maintenance Specialty

a. Basic Software Engineering Knowledge

Description This component includes basic knowledge about software engineering as a discipline. It covers basic knowledge about the fundamentals of computing, software product development, software management, and various software application domains. It embodies minimum prerequisite knowledge for more detailed and specialized study of software engineering.

Competency Requirements Engineers must have awareness of the existence and the context of the basic elements of software engineering. They must be able to provide a general, informal explanation of the terminology, concepts, and techniques associated with the discipline of computing; and they must be able to identify references that provide greater depth of knowledge about software engineering.

Rationale There are a wide variety of activities and tasks engaged in by software engineers that have responsibility for maintenance of software. The knowledge, skills, and capabilities required depend on the size and complexity of the software and the specific maintenance tasks. The chief responsibility for a "maintenance" software engineer is to analyze a maintenance request, identify alternative solutions, chose the best solution, and then design, implement, document, and test the solution. Next they may have to perform additional tests to verify that the specific solution did not generate any negative side-effects on the final product. The maintenance engineer may be involved throughout the life cycle of a software system (from working with acquisition to help insure a maintainable system is developed to assisting in the installation phase of the development). Hence, the engineer must have familiarity with an extensive array of basic computing and software engineering knowledge terms and concepts across the Software Product Engineering, Software Management, and Software Domains categories. In-depth knowledge of each area and unit is not required, but general awareness of the content and significance of each unit is important.

b. Computing Fundamentals

Description This component includes knowledge, concepts, theory, principles, methods, skills, and applications of computing that form the foundation for the development of software and the discipline of software engineering. Specifically it includes knowledge of algorithms and data structures, computer system organization and operation, communication and network essentials, various computer architectures, the fundamentals of operating systems, and discrete mathematics.

Competency Requirements Engineers must be able to explain, through definition and example, the terminology and concepts of computing fundamentals. They must be able to use their knowledge of computing fundamentals to monitor and evaluate the maintenance of software products.

Rationale A maintenance software engineer is required to develop software modules/patches that improve or fix existing functionality for operational software, and at times provide new functionality. Therefore, it is important that software engineers possess basic understanding of the fundamentals of computing.

c. Software Product Engineering

Description This component includes comprehensive knowledge about software requirements engineering, software design, software testing and software maintenance.

Competency Requirements Engineers must be able to apply knowledge about requirements, design, testing, and maintenance to maintain a software product. They must be able to inform others about the content and practices associated with software product engineering.

Rationale The maintenance software engineer must respond to a written request for some form of maintenance (correction, preventive, enhancement, etc.). This written request acts as a requirement for a special form of software development. The software engineer maintenance specialist is required to design, implement, and test a product that satisfies that requirement. Finally, the maintenance specialty area is responsible for testing the overall system (system and regression testing), and putting the new version of the product back into operation. Therefore, software engineers with maintenance specialties need to possess execution level knowledge in key areas of software product engineering.

d. Software Implementation

Description This component covers in-depth knowledge associated with software construction. It includes knowledge about algorithm design, detail design, data abstraction, information hiding, programming paradigms, coding, and unit, integration and system testing.

Competency Requirements Engineers must be able to apply knowledge about software construction to develop and maintain small modules. They must be

able to inform others about the content and practices associated with software construction.

Rationale In order to make changes to software, maintenance engineers must be proficient in software construction; they must have expert knowledge about detailed design, programming, and testing through the system level. Because of the nature of FAA software, such engineers need to be able to implement changes in software that involve human-computer interfaces, and real-time and distributed computing.

e. Software Maintenance Fundamentals

Description This component provides comprehensive coverage of software installation and maintenance. It includes knowledge about the maintenance process, maintenance operations, maintenance cost estimation, change and version control, and maintenance measurement and analysis.

Competency Requirements Engineers must be able to apply knowledge about software maintenance to maintain software products. They must be able to inform others about the content and practices associated with software maintenance.

Rationale Software maintenance engineers are required to perform all aspects of software maintenance at the FAA. As a result, the maintenance specialist must possess execution level knowledge about software maintenance operations and the software maintenance process.

f. Software Management

Description The component provides overview knowledge about software management responsibilities, methods, and activities. It includes basic knowledge about the terminology and techniques associated with project planning, quality assurance, software metrics, configuration management, risk management, and software processes.

Competency Requirements Engineers must be able to explain, through definition and example, software management terminology and concepts. They should be able to identify and evaluate management tasks associated with software maintenance.

Rationale The primary responsibility of the maintenance software engineer is to design, implement, and test software modules that will enhance and preserve the performance of an operational software product. This requires an understanding level familiarity with management concepts, methods, techniques, procedures and standards that are used for producing high-quality software products.

III. SOFTWARE ENGINEERING CURRICULUM SPECIFICATION

A. CURRICULUM ORGANIZATION

The basic components of the framework are a set of fourteen "learning modules" that have content and learning activities for conveying knowledge and practice that is embodied in the competency requirements in Section II. Table 1 lists and provide basic information about the fourteen modules. Detailed specification for the below listed modules is contained in part B of this section. These module specifications provide information, guidance, and recommendations that could be used to design and implement the module, or they could be used to evaluate and select appropriate equivalent existing training modules.

Table 1: Software Engineering Learning Modules

ID	Name	Competency Level	Coverage	Specialty
LM1	Basic Software Engineering Knowledge	Awareness	KC 2, 3, 4 *	A, T, M**
LM2	Computing Fundamentals	Understanding	KC 1	A, T, M
LM3	Software Domains	Understanding	KA 4.3, 4.6	A, T, M
LM4	Software Management (Fundamental)	Understanding	KA 3.1, 3.2, 3.3, 3.4, 3.5	T, M
LM5	Software Management (Advanced)	Execution	KA 3.1, 3.2, 3.3, 3.4, 3.5	A
LM6	Product Quality Control	Execution	KU 2.4.5, 2.5.1 KA 3.3	A, T
LM7	Software Product Engineering (Basic)	Understanding	KA 2.1 KU 2.2.1, 2.2.2, 2.2.3, 2.2.4, 2.3.1, 2.4.1	T
LM8	Software Product Engineering (Intermediate)	Understanding	KA 2.1 KU 2.2.1, 2.2.2, 2.2.3, 2.2.4, 2.4.2, 2.4.3, 2.4.4, 2.4.6, 2.4.7, 2.5.1, 2.5.2, 2.5.5	A, M
LM9	Software Product Engineering (Advanced)	Execution	KA 2.1, 2.2., 2.4	M
LM10	Software Implementation	Execution	KA 1.1, 2.3, 2.4 KU 1.5.2, 1.5.3, 2.2.5	M
LM11	Software Requirements	Execution	KU 2.1.1	A
LM12	Software Testing	Execution	KU 2.4.2, 2.4.3, 2.4.4, 2.4.5, 2.4.6, 2.4.7	T
LM13	Software Maintenance	Execution	KU 2.5.1, 2.5.2, 2.5.3	M
LM14	Software Acquisition	Execution	KA 3.6	A

* KC – knowledge category, KA – knowledge area, KU – knowledge unit

** A – Acquisition, T- Testing, M - Maintenance

B. LEARNING MODULE SPECIFICATIONS

This section contains detailed specifications for the fourteen modules listed in Table 1. Each module specification includes information about related software engineering knowledge, the competency level associated with the module, a module rationale, prerequisite requirements, a short module description, and a listing of the objectives and content of the module. Each specification also includes a recommended delivery format, suggested assessment techniques, and an estimation of required student effort. The last part of each specification describes appropriate resources and references for a module. Modules with an execution level of competency require additional experience as part of their training; the additional experiential training provides a deeper understanding and appreciation of the material needed to achieve the required competency.

A learning module specification presents a high-level design for appropriate training material that would support acquiring the specified knowledge at the indicated level of competency. These modules provide guidance for the implementation of software engineering training material. There are several factors that effect the depth of the coverage for a specific learning module: the SwE-BOK elements, the competency level, the duration and total student effort for the module, and the detail in which the module content is described. For example, for a module at an "awareness" competency level, with a high-level description of module content and a short duration, a very broad and high-level coverage of material would be appropriate. While a module at the "execution" competency level, with a detailed module content and a long duration, would require a more comprehensive coverage of the material with appropriate "execution" activities incorporated within the module training.

Appendix A includes information about each of the Learning Resources (LRs) sited in the learning module specifications. The recommended learning resources for each learning module are divided to two categories. Although they may not cover every LM topic, the primary LRs best satisfy the objectives of the LM; however, they may include additional material, be in a different format, and require different amounts of effort than that suggested in the module specification. The secondary LRs cover only a portion of the material described in the learning module. It is important to note that, the members of this project have not assessed the quality or effectiveness of the learning resources; therefore, siting of a specific LR does not signify its endorsement.

1. Basic Software Engineering Knowledge

Module LM 1	**Basic Software Engineering Knowledge**
Related SwE-BOK Elements	2 Software Product Engineering 3 Software Management 4 Software Domains
Competency Level	Awareness
Rationale	There are a wide variety of activities and tasks engaged in by FAA software engineers. The knowledge, skills, and capabilities required depend on the size and complexity of the software/system and the specific software-related tasks. The engineer must have familiarity with an extensive array of basic computing and software engineering knowledge terms and concepts across the Software Product Engineering, Software Management, and Software Domains categories. In-depth knowledge of each area and unit is not required, but general awareness of the content and significance of each category is important.
Prerequisite Knowledge	• No prerequisite knowledge

Description	The module provides an overview of software engineering as a discipline. It covers basic knowledge about software product development, software management, and various software application domains. It is designed for anyone new to software engineering. It provides minimum prerequisite knowledge for more detailed and specialized study of software engineering.
Module Objectives	Upon completion of this module the engineer will be able: • identify and discuss the technical and engineering activities of producing a software product • describe the concepts, methods, techniques, and procedures for managing software products and projects • characterize the knowledge in specific domains that involve computing and software engineering application or utilization.

Module Content	The module will provide a brief overview of each of the following topic areas:
	• Introduction to Software Engineering
	  ➢ Nature of Software
	  ➢ Software Crises
	• Software Product Engineering
	  ➢ Requirements Engineering
	  ➢ Design
	  ➢ Coding
	  ➢ Testing
	  ➢ Operation and Maintenance
	• Software Management
	  ➢ Project Management
	  ➢ Risk Management
	  ➢ Quality Management
	  ➢ Configuration Management
	  ➢ Process Management
	  ➢ Acquisition
	• Software Domains
	  ➢ Database Systems
	  ➢ Human-Computer Interaction
	  ➢ Real-Time Systems

Recommended Module Format and Learning Activities	• three hours of reading/study preparation
	• two day workshop
	• daily format
	  ➢ lectures and discussions on various software engineering topics
	  ➢ discuss a case study of the life-cycle of a real software product
	• in-class exercises – analyze the case study and answer a set of questions about software engineering concepts and terminology.

Required Effort	19 hours
Assessment	• pre and post workshop self-assessment • workshop exercise results
Resources	• Primary Resources: LR 2, LR 16, LR 21, LR 24, LR 43 • Secondary Resources: LR 1, LR 3, LR 4, LR 5, LR 44
References	• Brooks, F.P., "No Silver Bullet – Essence and Accident", *The Mythical Man-Month, Essays on Software Engineering, Anniversary Edition*, Addison-Wesley, 1995. • Dorfman, M. and Thayer, R., eds., *Software Engineering*, IEEE Computer Society Press, 1997. • Marciniak, John J., *Encyclopedia of Software Engineering*, John Wiley & Sons, Inc., 1994.

	• Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, Fourth Edition, McGraw-Hill, 1997. • Sommerville, I., *Software Engineering*, 5th Edition, Addison-Wesley, 1995.

2. Computing Fundamentals

Module LM 2	**Computing Fundamentals**
Related SwE-BOK Elements	1 Computing Fundamentals
Competency Level	Understanding
Rationale	In order for FAA software engineers to be able to interact effectively with contracted software, and to test and maintain the software acquired, it is necessary for them to have an understanding of fundamental computing terms and concepts. It is important that these engineers possess basic understanding of computer system organization and operation, communication and network essentials, various computer architectures, and the fundamentals of operating systems.
Prerequisite Knowledge	• No prerequisite knowledge
Description	• This module includes knowledge, concepts, and principles of computing that are essential to understanding of the development of software products. The module covers five major areas: algorithms and data structures, computer architecture, mathematical foundations, operating systems, and programming languages.
Module Objectives	Upon completion of this module the engineer will be able: • recognize and discuss the key topics and terms within computing • describe the relationship and dependency between the different areas of computing • explain how the various areas of computing relate to software development • pursue more advanced study in computing

Module Content	The module will provide an overview of each of the following topics:
	• Algorithms and Data Structures
	  ➢ Basic Data Structures
	  ➢ Design and Analysis of Algorithms
	• Computer Architecture
	  ➢ Digital Systems
	  ➢ Computer System Organization
	  ➢ Communications and Networks
	• Mathematical Foundations
	  ➢ Discrete Mathematics
	  ➢ Probability and Statistics
	• Operating Systems
	  ➢ Operating Systems Fundamentals
	  ➢ Process Management and Memory Management
	  ➢ Security and Protection
	  ➢ Distributed and Real-time Systems
	• Programming Languages
	  ➢ Theory of Programming Languages
	  ➢ Programming Paradigms
	  ➢ Programming Language Implementation

Recommended Module Format and Learning Activities	• five hours of reading/study preparation • five day workshop with one day devoted to each of the five major topic areas • each one day segment is independent of the others; hence, engineers with knowledge about some of the segments could take just those segments needed • daily format ➢ lectures and discussions on various computing topics ➢ short in-class individual exercises on the computing topics being discussed
Required Effort	50 hours
Assessment	• pre and post workshop self-assessment • workshop exercise results
Resources	• Primary Resources: LR 9, LR 10, LR 11, LR 13 • Secondary Resources: LR 6, LR 7, LR 8, LR 12, LR 14, LR 15
References	• Brookshear, J. Glenn, *Computer Science: An Overview*, 4th edition, Benjamin/Cummings, 1994. • Grimaldi, R., *Discrete and Combinatorial Mathematics*, 3rd edition, Addison-Wesley, 1994. • Guttman, I., Wilkes, S. and Hunter, J., *Introductory Engineering Statistics*, 2nd edition, John Wiley, 1971. • Mano, M., *Computer System Architecture*, 3rd edition, Prentice-Hall, 1993. • Tanenbaum, A., *Operating System Design and Implementation*, Prentice-Hall, 1987. • Weiss, M., *Data Structures and Algorithm Analysis in C++*, Addison, Wesley, 1994. • Wilson, Leslie B. and Robert G. Clark. *Comparative Programming Languages*, 1993.

3. Software Domains

Module LM 3	**Software Domains**
Related SwE-BOK Elements	4.3 Human-Computer Interaction 4.6 Real-Time Systems
Competency Level	Understanding
Rationale	Because of the nature of FAA computing requirements, intensive user interaction and real-time embedded computing are typical characteristics of its software systems. Hence, it is important that ARA software engineers have familiarity and understanding in other "specialty" software domains; in particular, the engineer should have knowledge and understanding of human computer interaction and real-time systems.
Prerequisite Knowledge	Completion of the following module (or equivalent knowledge): • LM 1: Basic Software Engineering • LM2: Computing Fundamentals
Description	The module is divided into two sub-modules: one in human computer interaction and one in real-time systems. The human computer interaction part covers user interfaces, computer graphics, and hypertext/hypermedia. The real-time systems part includes knowledge about basic properties of real-time application software and the development of real-time software systems.

Module Objectives	Upon completion of this module the engineer will be able: • recognize and discuss the key topics and terms for human computer interaction • recognize and discuss the key topics and terms for real-time systems • explain the principal issues in developing software that involves human computer interaction • explain the principal issues in developing software that involves real-time systems
Module Content	The module will provide a overview of each of the following topics: • Human-Computer Interaction ➢ User Interfaces ➢ Computer Graphics ➢ Hypertext/Hypermedia • Real-Time Systems ➢ Basic Properties of Real-time Application Software ➢ Design and Implementation for Real-time Software ➢ Concurrent Programming ➢ Resource Management ➢ Real-time Programming Languages and Operating Systems.
Recommended Module Format and Learning Activities	• four hours of reading/study preparation • two day workshop with one day devoted to each of the two major topic areas • each day is treated as sub-module that is independent of the other • daily format ➢ lectures and discussions on various major topics of the day • short in-class individual exercises on the topics being discussed
Required Effort	20 hours
Assessment	• pre and post workshop self-assessment • workshop exercise results
Resources	• Primary Resources: LR 16, LR 21 • Secondary Resources: LR 17, LR 18, LR 19, LR 20
References	• Barfield, L., *The User Interface: Concepts and Design*, Addison-Wesley, 1993. • Burns, A. and Wellings, A., *Real-Time Systems and their Programming Languages*, Addison-Wesley, 1997. • Gomaa, H., *Software Design Methods for Concurrent and Real-Time Systems*, Addison-Wesley, 1993. • Hill, F., *Computer Graphics*, Macmillan, 1990. • Proctor, R. and Zandt, T., *Human Factors*, Allyn and Bacon, 1994.

4. Software Management (Fundamental)

Module LM 4	**Software Management (Fundamental)**
Related SwE-BOK Elements	3.1 Software Project Management 3.2 Software Risk Management 3.3 Software Quality Management 3.4 Software Configuration Management 3.5 Software Process Management
Competency Level	Understanding
Rationale	In order to acquire, test and maintain software, FAA software engineers require an understanding level familiarity with management concepts, methods, techniques, procedures and standards that are used for

	producing high-quality software products. Hence, such engineers must have a good understanding of project planning, quality assurance, software metrics, configuration management, risk management, and software processes.
Prerequisite Knowledge	Completion of the following module (or equivalent knowledge): • LM 1: Basic Software Engineering
Description	The module gives an overview of software management responsibilities, methods, and activities. The module provides basic knowledge about the terminology and techniques associated with project planning, quality assurance, software metrics, configuration management, risk management, and software processes.
Module Objectives	Upon completion of this module the engineer will be able: • identify and discuss the key operations and issues associated with software management • describe the principal elements of project planning, quality assurance, software metrics, configuration management, and risk management • discuss the need for software process and identify common software process improvement issues
Module Content	The module will provide a brief overview of each of the following topic areas: • Software Project Management • Software Risk Management • Software Quality Management • Software Configuration Management • Software Process Management
Recommended Module Format and Learning Activities	• three hours of reading/study preparation • two day workshop • daily format ➢ lectures and discussions on various software management topics ➢ discuss a case study of the life-cycle of a real software product • in-class exercises – analyze the case study and answer a set of questions about software management concepts and terminology.
Required Effort	19 hours
Assessment	• pre and post workshop self-assessment • workshop exercise results
Resources	• Primary Resources: LR 22, LR 25 • Secondary Resources: LR 23, LR 24
References	• Dorfman, M. and Thayer, R., eds., *Software Engineering*, IEEE Computer Society Press, 1997. • Marciniak, John J., *Encyclopedia of Software Engineering*, John Wiley & Sons, Inc., 1994. • Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, Fourth Edition, McGraw-Hill, 1997. • Sommerville, I., *Software Engineering*, 5th Edition, Addison-Wesley, 1995.

5. Software Management (Advanced)

Module LM 5	**Software Management (Advanced)**
Related SwE-BOK Elements	3.1 Software Project Management 3.2 Software Risk Management 3.3 Software Quality Management

	3.4 Software Configuration Management 3.5 Software Process Management
Competency Level	Execution
Rationale	The primary role of acquisition software engineers is to monitor the development/acquisition of a software system. For many FAA systems the size, complexity, and criticality of the software components require significant and long-term management oversight by the acquisition engineer. They must be able to review a software development plan, evaluate a project budget, monitor project progress, participate in a software review, and study and analyze software measurement data. This requires knowledge about a wide range of software management activities: project planning, quality assurance, software metrics, configuration management, risk management, and software processes.
Prerequisite Knowledge	Completion of the following module (or equivalent knowledge): • LM 1: Basic Software Engineering • FAA Course: Project Management for Software Intensive Systems
Description	The module provides for a comprehensive study of the concepts, methods and techniques for managing the development of software products. It includes learning about software activities concerned with project management, management of risk, the configuration of a software system, knowledge about how to produce high-quality software and the development and improvement of software processes.
Module Objectives	Upon completion of this module the engineer will be able: • develop software project plan • perform a software risk analysis • assess a configuration management system • evaluate the quality management organization and activities for a software project • describe the key elements and issues in software process improvement

Module Content	The module will cover the following topic areas:
	• Software Project Management
	• Software Risk Management
	• Software Quality Management
	  ➢ Software Quality Assurance
	  ➢ Verification and Validation
	  ➢ Software Metrics
	• Software Configuration Management
	  ➢ Software Configuration Identification
	  ➢ Software Configuration Control
	  ➢ Software Configuration Audit and Status Accounting
	• Software Process Management
	  ➢ Quantitative Software Process Management
	  ➢ Software Process Improvement
	  ➢ Software Process Assessment
	  ➢ Software Process Engineering

Recommended Module Format and Learning Activities	• five hours of reading/study preparation
	• four day workshop
	• daily format
	  ➢ Morning – lectures and discussions
	  ➢ Afternoon – individual and group exercises
	• exercises
	  ➢ develop a software project plan
	  ➢ develop a risk plan

	➢ assess a configuration management system ➢ assess a quality management plan ➢ study and report on an organization-level software process system
Required Effort	40 hours
Additional Training	• 6 months of supplemental experiential learning involving software management activities
Assessment	• Pre and post workshop self-assessment • Workshop exercise results
Resources	• Primary Resources: LR 22, LR 25, LR 26, LR 29, LR 30, LR 35, LR 36 • Secondary Resources: LR 23, LR 24, LR 27, LR 28,LR 31, LR 32, LR 33, LR 34, LR 37, LR 38
References	• Buckley, F. J., *Implementing Configuration Management: Hardware, Software, and Firmware*, IEEE Computer Society Press, 1996.Gillies, A. C., Software Quality*: Theory and Management*, Chapman & Hall, 1992. • Hall, E. M., *Managing Risk*, Addison-Wesley, 1998. • Thayer, R. H., *Software Engineering Project Management: A Top-Down View*, *Tutorial: Software Engineering Project Management*, IEEE Computer Society Press, 1988. • Humphrey, W. S., *Managing the Software Process*, Addison-Wesley, 1989. • Dorfman, M. and Thayer, R., eds., *Software Engineering*, IEEE Computer Society Press, 1997. • Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, Fourth Edition, McGraw-Hill, 1997.

6. Product Quality Control

Module LM 6	**Product Quality Control**
Related SwE-BOK Elements	2.5.1 Software Installation and Operation 3.3.1 Software Quality Assurance 3.3.2 Verification and Validation 3.3.3 Software Metrics
Competency Level	Execution
Rationale	The primary responsibility of a test software engineer is to validate and verify the quality of the software product. This requires familiarity with concepts, methods, techniques, procedures and standards for producing high-quality software products; and execution knowledge about quality planning and control, verification and validation activities, measurement of product and process attributes, and assessing effective software operation.
Prerequisite Knowledge	Completion of the following modules (or equivalent knowledge): • LM 1: Basic Software Engineering • LM 2: Computing Fundamentals • LM 7: Software Product Engineering (Basics)
Description	This module provides in-depth knowledge and activities that are necessary for producing high quality software. It covers material on quality assurance, formal review, software metrics, and assessment and analysis of software quality.
Module Objectives	Upon completion of this module the engineer will be able: • describe concepts, methods, techniques, procedures and standards for producing high-quality software products

	• explain how quality planning and control techniques are used in software development • use verification and validation activities to produce quality software • measure product and process attributes • use software metrics to assess process and product quality • assess effective software operation
Module Content	The module will include the following topic areas: • Quality Assurance ➢ Organization of Quality Assurance Units ➢ Quality Control (planning, oversight, record keeping, analysis, auditing, and reporting) ➢ Quality Assurance Techniques (Pareto analysis, trend analysis, statistical quality control, and regression testing) • Validation and Verification ➢ Basic Verification and Validation (V&V) Concepts (methods, activities, and deliverables associated with each phase in the software life cycle.) ➢ V&V Planning and Organization ➢ V&V Techniques (personal reviews, walkthroughs, and inspections; traceability analysis; formal verification techniques; Cleanroom techniques, and software testing) • Software Reliability Model • Software Fault Tree Analysis • Software Metrics and Measurements ➢ Software Metric Fundamentals (collection, computation, analysis and feedback) ➢ Metric Classification (product metrics, resource metrics, process metrics) • Methods and Techniques for Installing a Software Product • Documentation and Transition to System Operation
Recommended Module Format and Learning Activities	• six hours of reading/study preparation • five day workshop delivered one day a week • daily format ➢ morning – lectures and discussions ➢ afternoon – individual and group in class exercises, plus individual take home exercises • exercises ➢ inspection of a software design specification ➢ individual code review ➢ computation and analysis of a set of metrics based on data collected from a software development effort (requirements and design metrics, inspection/review data, testing data) ➢ analysis of a software quality assurance plan ➢ review of software installation/operation documentation
Required Effort	46 hours
Additional Training	• 6 months of supplemental experiential learning involving software quality control activities
Assessment	• pre and post workshop self-assessment • workshop exercise results
Resources	• Primary Resources: LR 30, LR 39, LR 42 • Secondary Resources: LR 29, LR 31, LR 32, LR 41
References	• 1012-1998, *IEEE Standard for Software Verification and Validation*. • 730-1998, *IEEE Standard for Software Quality Assurance Plans*. • MIL-S-52779A, *Software Quality Assurance Program Requirement,*

	1979. • DI-QCIC-80572, *DOD Software Quality Program Plan*, 1988. • 1061-1998, *IEEE Standard for a Software Quality Metrics Methodology*. • Arthur, L.J., *Software Evolution*, John Wiley, 1988. • Gillies, A. C., *Software Quality: Theory and Management*, Chapman & Hall, 1992. • Weinberg, *Quality Software Management (Volume 1-3)*, Dorset House Publishing, 1993. • Deutsch & Willis, *Software Quality Engineering, A Total Technical and Management Approach*, Prentice Hall, 1988. • Kaplan, *Secrets of Software Quality*, McGraw Hill, 1995. • Ebenau, R., *Software Inspection Process*, McGraw Hill, 1994. • Ince, D., *ISO 9001 and Software Quality Assurance*, McGraw-Hill, 1994. • Kan, S. H., *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 1995.

7. Software Product Engineering (Basic)

Module LM 7	**Software Product Engineering (Basics)**
Related SwE-BOK Elements	2.1 Software Requirement Engineering 2.2.1 Architectural Design 2.2.2 Abstract Specification 2.2.3 Interface Design 2.2.4 Data Structure Design 2.3.1 Code Implementation 2.4.1 Unit Testing
Competency Level	Understanding
Rationale	Since the primary responsibility of a software engineer with testing responsibility is to monitor and verify the quality of a software product , it is important for such engineers to have a good comprehension of the activities associated with software product development. In particular, the engineer must have a sound understanding of requirements engineering, design specification (including software architectures, interface design, and data design), software coding (on occasion the test engineer may have to prepare test drivers and other elements of the test environment), and a good understanding of unit testing. The engineer may work with a contractor as part of a development team, monitoring their testing activity. As such they must be able to read, comprehend, and evaluate the quality of requirement specifications and high-level design documents; and understand the role of requirements tracing to ensure all requirements have been properly implemented.
Prerequisite Knowledge	Completion of the following modules (or equivalent knowledge): • LM 1: Basic Software Engineering • LM 2: Computing Fundamentals • LM 3: Software Domains • LM 4: Software Management (Fundamentals)
Description	This module provides a general overview of software requirements engineering, software design, software construction, and the software testing of a small software component.
Module Objectives	Upon completion of this module the engineer will be able: • describe concepts, methods, techniques, procedures and standards

	associated with software requirements and design • track software requirements throughout the software development life cycle • perform a trace of an algorithm for a small software component • assess the effectiveness of the implementation and test of a small software component
Module Content	The module will include the following topic areas: • Software Requirements Engineering (requirement elicitation, analysis, and specification) • Software Design (Abstract specification, architectural, interface and data structure design) • Software Implementation (Algorithm development, and modular and incremental programming)
Recommended Module Format and Learning Activities	• three hours of reading/study preparation • three days workshop • daily format ➢ Morning – lectures and discussions ➢ Afternoon – individual and group exercises • exercises ➢ Requirement analysis ➢ Functional decomposition ➢ Requirement tracking ➢ Trace algorithm ➢ Evaluate software implementation
Required Effort	27 hours
Assessment	• Pre and post workshop self-assessment • Workshop exercise results
Resources	• Primary Resources: LR 43, LR 44, LR 45 • Secondary Resources: LR 46, LR 48, LR 49, LR 51
References	• 830-1993, *IEEE Recommended Practice for Software Requirement Specifications*. • P1233/D3, *IEEE Guide for Developing System Requirements Specification*. • Gause, D.C, G.M. Weinberg, *Exploring Requirements Quality Before Design*, Dourset House Pub., 1989. • Davis, A., *Software Requirements: Objects, Functions & States*, Prentice Hall, 1993. • Hetzel, B., *The Complete Guide to Software Testing*, Wiley, 1988 • Kendall, K.E., and J.E. Kendall, *Systems Analysis and Design*, Prentice Hall, 1992. • Pressman, R.S. *Software Engineering A Practitioner's approach*, McGraw-Hill, 1997. • Budgen, D. *Software Design*, Addison wesley, 1994.

8. Software Product Engineering (Intermediate)

Module LM 8	**Software Product Engineering (Intermediate)**
Related SwE-BOK Elements	2.2 Software Requirement Engineering 2.2.5 Architectural Design 2.2.6 Abstract Specification 2.2.7 Interface Design 2.2.8 Data Structure Design 2.4.1 Unit Testing

	2.4.2 Integration Testing 2.4.3 System Testing 2.4.4 Performance Testing 2.4.6 Installation Testing 2.4.7 Test Documentation 2.5.1 Software Installation and Operation 2.5.2 Software Maintenance Operation 2.5.5 Software Reengineering
Competency Level	Understanding
Rationale	Since the primary responsibility of an acquisition software engineer is to monitor and support the development of a software product it is important for such engineers to have a good comprehension of the activities associated with software product development. In addition, they should have sufficient knowledge to effectively judge and help avoid the maintainability and evolution problems. This requires a sound understanding of requirements engineering, design specification (including software architecture, interface and data design), and software testing beyond the unit level. Also, in the beginning of a project the acquisition engineer must interact with system engineering in order to insure that the proposed software development activities are achievable. Since acquisition software engineers have to make judgements about the viability and effectiveness of the contractor development methods and processes, they need to be able to read, comprehend, and evaluate the quality of requirements specifications, high-level design documents, and system test plans. To insure good quality, they must also be able to trace requirements through all phases of the software development life cycle.
Prerequisite Knowledge	Completion of the following modules (or equivalent knowledge): • LM 1: Basic Software Engineering • LM 2: Computing Fundamentals • LM 3: Software Domains • LM 4: Software Management (Fundamentals)
Description	This module provides a detailed overview of software product engineering. In particular, it covers knowledge about the terms, methods, and techniques used in software requirements engineering, software design, software testing, and installation and maintenance of software.
Module Objectives	Upon completion of this module the engineer will be able: • describe concepts, methods, techniques, procedures and standards associated with software requirements and design • track software requirements through software development life cycle • describe the process, techniques and outcomes of software testing • discuss the issues associated with software installation and maintenance
Module Content	The module will include the following topic areas: • Software Requirements Engineering (requirement elicitation, analysis and specification) • Software Design (Abstract specification, architectural, interface, and data structure design) • Software Testing (Test documentation, integration, system, performance, and installation testing) • Software Installation and Maintenance
Recommended Module Format and Learning Activities	• five hours of reading/study preparation • four days workshop • daily format ➢ Morning – lectures and discussions

	➢ Afternoon – individual and group exercises • exercises ➢ Requirement analysis ➢ Functional decomposition ➢ Requirement tracking ➢ Algorithm design ➢ Software implementation
Required Effort	37 hours
Assessment	• Pre and post workshop self-assessment • Workshop exercise results
Resources	• Primary Resources: LR 43, LR 44, LR 45, LR 60, LR 68 • Secondary Resources: LR 46, LR 48, LR 49, LR 51,LR 59, LR 65
References	• 830-1993 *IEEE Recommended Practice for Software Requirement Specifications*. • P1233/D3 *IEEE Guide for Developing System Requirements Specification*. • 829-1998 *IEEE Standard for Software Test Documentation.* • Gause, D.C, G.M. Weinberg, *Exploring Requirements Quality Before Design*, Dourset House Pub., 1989. • Davis, A., *Software Requirements: Objects, Functions & States*, Prentice Hall, 1993. • Hetzel, B., *The Complete Guide to Software Testing*, Wiley, 1988. • Beizer, B., *Software Testing Techniques*, Thompson Computer Press, 1990. • Jorgensen, P.C., *Software Testing A Craftsman's Approach*, CRC, 1995. • Kendall, K.E., and J.E. Kendall, *Systems Analysis and Design*, Prentice Hall, 1992. • Pressman, R.S. *Software Engineering A Practitioner's approach*, McGraw-Hill, 1997. • Budgen, D. *Software Design*, Addison wesley, 1994. • Glass, R.L., R.A. Noiseux, *Software Maintenance Guidebook*, Prentice Hall, 1981.

9. Software Product Engineering (Advanced)

Module LM 9	**Software Product Engineering (Advanced)**
Related SwE-BOK Elements	2.3 Software Requirement Engineering 2.4 Software Design 2.4 Software Testing
Competency Level	Execution
Rationale	The software engineer with the maintenance specialty is responsible to respond to a written request for some form of maintenance (correction, preventive, enhancement, etc.). This written request acts as a requirement for a special form of software development. The software engineer maintenance specialist may be required to design, implement, and test a product that satisfies that requirement. Finally, the maintenance specialty area is responsible for testing the overall system (system and regression testing), and putting the new version of the product back into operation. Therefore, software engineers with maintenance specialties need to possess execution level knowledge in key areas of software product engineering.

Prerequisite Knowledge	Completion of the following modules (or equivalent knowledge): • LM 1: Basic Software Engineering • LM 2: Computing Fundamentals • LM 3: Software Domains • LM 4: Software Management (Fundamentals) • LM 8: Software Product Engineering (Intermediate)
Description	This module provides in-depth coverage of software product engineering. In particular, it covers the knowledge and activities for methods and techniques used in software requirements engineering, software design, and software testing.
Module Objectives	Upon completion of this module the engineer will be able: • describe concepts, methods, techniques, procedures and standards associated with software requirements and design • track software requirements through software development life cycle • understand the process, techniques and outcomes of the software testing • understand issues associated with software installation and maintenance operation
Module Content	The module will include the following topic areas: • Software Requirements Engineering ➢ Requirement Elicitation ➢ Requirement Analysis ➢ Requirement Specification • Software Design ➢ Architectural Design ➢ Abstract Specification ➢ Interface Design ➢ Data Structure Design ➢ Algorithm Design • Software Testing ➢ Unit Testing ➢ Integration Testing ➢ System Testing ➢ Performance Testing ➢ Installation Testing ➢ Documentation
Recommended Module Format and Learning Activities	• ten hours of reading/study preparation • five days workshop • daily format ➢ Morning – lectures and discussions ➢ Afternoon – individual and group exercises • exercises ➢ Requirement analysis ➢ Functional decomposition ➢ Requirement tracking ➢ Algorithm design ➢ Software implementation ➢ Software maintenance ➢ Software reengineering
Required Effort	50 hours
Additional Training	• 6 months of supplemental experiential learning involving product engineering activities
Assessment	• Pre and post workshop self-assessment • Workshop exercise results

Resources	• Secondary Resources: LR 47, LR 50, LR 64
References	• 830-1993 *IEEE Recommended Practice for Software Requirement Specifications.* • P1233/D3 *IEEE Guide for Developing System Requirements Specification.* • 829-1998 *IEEE Standard for Software Test Documentation.* • Gause, D.C, G.M. Weinberg, *Exploring Requirements Quality Before Design*, Dourset House Pub., 1989. • Davis, A., *Software Requirements: Objects, Functions & States*, Prentice Hall, 1993. • Hetzel, B., *The Complete Guide to Software Testing*, Wiley, 1988. • Beizer, B., *Software Testing Techniques*, Thompson Computer Press, 1990. • Jorgensen, P.C., *Software Testing A Craftsman's Approach,* CRC, 1995. • Kendall, K.E., and J.E. Kendall, *Systems Analysis and Design*, Prentice Hall, 1992. • Pressman, R.S. *Software Engineering A Practitioner's approach*, McGraw-Hill, 1997. • Budgen, D. *Software Design*, Addison-Wesley, 1994. • Glass, R.L., R.A. Noiseux, *Software Maintenance Guidebook*, Prentice Hall, 1981. • Martin, J, and G. McClure*, Software Maintenance: The Problem and Its Solution*, Prentice Hall, 1983. • Parikh, G., N. Zvegintzov, *Tutorial on Software Maintenance*, IEEE Computer Society, 1983.

10. Software Implementation

Module LM 10	**Software Implementation**
Related SwE-BOK Elements	1.1 Algorithms and data structure 2.3 Software Coding 2.4 Software Testing 1.5.2 Programming paradigms 1.5.3 Programming language design and implementation 2.2.5 Algorithm design
Competency Level	Execution
Rationale	In order to make changes to software, maintenance engineers must be proficient in software construction; they must have expert knowledge about detailed design, programming, and testing through the system level. Because of the nature of FAA software, such engineers need to be able to implement changes in software that involve human-computer interfaces, and real-time and distributed computing.
Prerequisite Knowledge	Completion of the following modules (or equivalent knowledge): • LM 1: Basic Software Engineering • LM 2: Computing Fundamentals • LM 3: Software Domains • LM 9: Software Product Engineering (Advance)
Description	This module provides in-depth knowledge associated with software construction. The main purpose of this module is to provide information (such as processes, techniques and tools) that is necessary for a software engineer to start with a set of requirements, and use them to construct a software component that satisfies those requirements. Issues such as

	algorithm design, detail design, data abstraction, information hiding, unit, integration and system testing are be discussed in this module. In addition, this module provides a comparison between the different programming paradigms and languages.
Module Objectives	Upon completion of this module the engineer will be able: • describe concepts, methods, techniques, procedures and standards associated with software construction • track software requirements through software development life cycle • develop an algorithm from specification • develop a detail design, and implement that design using an appropriate programming language and paradigm • perform unit integration and system testing
Module Content	The module will include the following topic areas: • Software Implementation • Requirement Analysis • Algorithm Design • Programming • Testing (unit, integration, and system) • Programming Languages • Different Programming Paradigms • Different Programming Languages
Recommended Module Format and Learning Activities	• thirty hours of reading/study preparation • six days workshop (Two days during the first week, followed by one day for each of the next following four weeks) • daily format The first day of workshop will be all lecture, the next five days will have the following format ➢ Morning – lectures and discussions ➢ Afternoon – individual and group exercises • Exercises ➢ Number of programming assignments ➢ Unit and integration testing
Required Effort	78 hours
Additional Training	• 6 months of supplemental experiential learning involving software implementation
Assessment	• Pre and post workshop self-assessment • Workshop exercise results
Resources	• Primary Resources: LR 7 • Secondary Resources: LR 59
References	• Pressman, R.S. *Software Engineering A Practitioner's approach*, McGraw-Hill, 1997. • Savitch, W., *Problem Solving with C++ The Object of Programming*, Addison Wesely, 1996. • Carrano, Helman, Veroff, *Data Abstraction and Problem Solving with C++*, Addison-Wesley, 1998. • Dijkstra, E., *A Discipline of Programming*, Prentice Hall, 1976.

11. Software Requirements

Module LM 11	Software Requirements
Related SwE-BOK Elements	2.1.1 Requirement Elicitation
Competency Level	Execution
Rationale	The acquisition software engineer provides a critical interface between the FAA as a user/customer and the contractor as the developer; he/she must support the effort to establish a common understanding of the requirements to be addressed by a software product. Hence, it is important for the engineer to have knowledge and experience with methods and techniques for uncovering, discovering, and communicating functional and non-functional requirements and constraints. In addition, the acquisition engineer should have knowledge about the modeling of software requirements in the information, functional, data, and behavioral domains of a problem. An acquisition engineer must be equipped to make a trade-off analysis between functional requirements and the constraints on a system, along with all the derived requirements of a system, which highlight the affect on development cost and schedule.
Prerequisite Knowledge	Completion of the following modules (or equivalent knowledge): • LM 1: Basic Software Engineering • LM 2: Computing Fundamentals • LM 3: Software Domains • LM 8: Software Product Engineering (Intermediate)
Description	This module provides knowledge that supports the systematic development of a complete understanding of the problem domain. This unit also includes knowledge about methods and techniques for uncovering, discovering and communicating functional and non-functional requirements and constraints; it provides a foundation for decomposing a problem into intellectually manageable pieces by using objects, functions and states.
Module Objectives	Upon completion of this module the engineer will be able: • identify all the software stakeholders • discover software requirements • use appropriate interview types and techniques for requirement elicitation • use appropriate steps to assure the quality of the requirements
Module Content	The module will include the following topic areas: • Requirement Elicitation Techniques • Interview Techniques • Interview Types • Use Case Analysis • Viewpoint Analysis • Modeling • Partitioning • Feasibility Analysis • Requirements Quality Assurance
Recommended Module Format and Learning Activities	• four hours of reading/study preparation • two days workshop • daily format ➢ Morning – lectures and discussions ➢ Afternoon – individual and group exercises

	• exercises ➢ Number of Case studies and role playing
Required Effort	20 hours
Additional Training	• 3 months of supplemental experiential learning involving software requirements elicitation
Assessment	• Pre and post workshop self-assessment • Workshop exercise results
Resources	• Primary Resources: LR 52, LR 53, LR 56, LR 57 • Secondary Resources: LR 54, LR 55
References	• 830-1993 *IEEE Recommended Practice for Software Requirement Specifications*. • P1233/D3 *IEEE Guide for Developing System Requirements Specification.* • Pressman, R.S., *Software Engineering A Practitioner's Approach*, McGraw Hill, 1997. • Thayer, H.T., and M. Durfman, *System and Software Requirements Engineering*, IEEE Computer Society Press Tutorial, 1990. • Thayer, H.T., and M. Durfman, *Software Requirement Engineering*, IEEE Computer Society, 1997.

12. Software Testing

Module LM 12	**Software Testing**
Related SwE-BOK Elements	2.4.2 Integration Testing 2.4.3 System Testing 2.4.4 Performance Testing 2.4.5 Acceptance Testing 2.4.6 Installation Testing 2.4.7 Test Documentation
Competency Level	Execution
Rationale	Test engineers need both deep and broad knowledge about software testing; they are involved in a multi-stage process that consists of activities for validating the software product, from the most primitive elements up to a fully integrated system. Therefore, test engineers must have execution knowledge of test documentation, and integration, system, performance, acceptance, and installation testing.
Prerequisite Knowledge	Completion of the following modules (or equivalent knowledge): • LM1: Basic Software Engineering • LM2: Computing Fundamentals • LM4: Software Management (Fundamentals) • LM7: Software Product Engineering (Basics)
Description	This module provides in-depth knowledge and activities that are associated with software testing. It covers material on the software testing life cycle, testware development, software testing techniques and methods, and software testing metrics.
Module Objectives	Upon completion of this module the engineer will be able: • describe concepts, methods, techniques, procedures and standards associated with software testing • describe the software testing process and life cycle • differentiate between the different software testing techniques and methods • develop a test plan with the assurance of full coverage, using different

	testing techniques • identify test conditions and design test cases • implement a test plan to test a software product • apply the international IEEE testing standard • discuss the advantages and limitations of existing software testing tools • use software metrics to assess the software testing process and product quality
Module Content	The module will include the following topic areas: • Testing Life-cycle (planning, design, implementation, execution, analysis, and maintenance) • Test Documentation (test plan, test case specification, test script, test case, test log, test incident report, and test library) • Testing Levels (unit, integration, system) • Testing Methods and Techniques ➢ Functional Testing ➢ Black-box Testing ➢ White-box Testing ➢ Acceptance Testing ➢ Installation Testing ➢ Stress Testing ➢ Performance Testing ➢ Regression Testing • Test Case Identification and Design • Requirement Phase Testing • Design Phase Testing • Test Metrics ➢ Test Metrics Fundamentals (collection, computation, analysis and feedback) ➢ Test Metrics Classification (effort, evaluation, and execution) ➢ Test and Defect Tracking
Recommended Module Format and Learning Activities	• ten hours of reading/study preparation • two five day workshops (The first workshop, delivered during one week, presents the fundamentals, and the second workshop, delivered one day a week for five weeks, presents advance topics) • daily format ➢ Morning – lectures and discussions ➢ Afternoon – individual and group exercises • exercises ➢ Test plan generation ➢ Test case and testing technique identification ➢ Test case design ➢ Requirement testing ➢ Design testing ➢ Test and defect tracking ➢ Test metrics collection and analysis
Required Effort	90 hours
Additional Training	• 6 months of supplemental experiential learning involving software testing
Assessment	• Pre and post workshop self-assessment • Workshop exercise results
Resources	• Primary Resources: LR 38, LR 60, LR 61 • Secondary Resources: LR 58, LR 59, LR 62, LR 63
References	• 829-1998 *IEEE Standard for Software Test Documentation*.

	• 1465-1998 *IEEE Standard adaptation of ISO/IEC 12119-Software packages-Quality Requirements and Testing*. • 1012-1998 *IEEE Standard for Software Verification and Validation*. • 730-1998 *IEEE Standard for Software Quality Assurance Plans*. • 1061-1998 *IEEE Standard for a Software Quality Metrics Methodology*. • B. Hetzel, *A Complete Guide to Software Testing*, QED Information Sciences, 1984. • B. Beizer, *Software Testing Techniques*, Thompson Computer Press, 1982. • Roper M., *Software Testing*, McGraw-Hill, 1994. • W. Perry, *Effective Methods for Software Testing*, Wiley, 1995. • Jorgenson, *Software Testing, A Craftsman's Approach*, CRC, 1995.

13. Software Maintenance

Module LM 13	**Software Maintenance**
Related SwE-BOK Elements	2.5.1 Software Installation and Operation 2.5.2 Software Maintenance Operation 2.5.3 Software Maintenance Process
Competency Level	Execution
Rationale	Software maintenance engineers are required to perform all aspects of software maintenance at the FAA. As a result, the maintenance specialist must possess mastery level knowledge about software installation and maintenance, software maintenance operations, and the software maintenance process.
Prerequisite Knowledge	Completion of the following modules (or equivalent knowledge): • LM 1: Basic Software Engineering • LM 2: Computing Fundamentals • LM 3: Software Domains • LM 4: Software Management
Description	This module provides an in-depth coverage of software installation and maintenance. It covers terms, techniques and methods associated with the maintenance process, maintenance data analysis, and maintenance operations.
Module Objectives	Upon completion of this module the engineer will be able: • differentiate between different maintenance types • measure and improve the maintainability of software • estimate resources and cost of the maintenance • establish, enforce, and follow a change and version control procedure
Module Content	The module will include the following topic areas: • Software Maintenance Types • Maintainability Measurement • Maintenance Management • Software Maintenance Cost Estimation • Change Management Procedure • Configuration Management and Version Control in Maintenance

Recommended Module Format and Learning Activities	• five hours of reading/study preparation
	• two days workshop
	• daily format
	  ➢ Morning – lectures and discussions
	  ➢ Afternoon – individual and group exercises

	• exercises ➢ Maintenance case studies
Required Effort	21 hours
Additional Training	• 3 months of supplemental experiential learning involving software maintenance activities
Assessment	• Pre and post workshop self-assessment • Workshop exercise results
Resources	• Primary Resources: LR 65, LR 68 • Secondary Resources: LR 64, LR 66, LR 67
References	• Schach, S. *Software Engineering*, Akson Associate, 1990. • Lientz, B.P., E.B. Swanson, *Software Maintenance Management*, Addison-Wesley, 1980. • Boehm, B.W., *The economics of software maintenance*, Proc. Software maintenance workshop, 1983. • Glass, R.L., R.A. Noiseux, *Software Maintenance Guidebook*, Prentice Hall, 1981. • Martin, J, and G. McClure*, Software Maintenance: The Problem and Its Solution,* Prentice Hall, 1983. • Parikh, G., N. Zvegintzov, *Tutorial on Software Maintenance*, IEEE Computer Society, 1983. • *A Review of Software Maintenance Technology*, Rome Air Development Center.

14. Software Acquisition

Module LM 14	**Software Acquisition**
Related SwE-BOK Elements	3.6 Software Acquisition
Competency Level	Execution
Rationale	A software engineer working in the acquisition area must have explicit knowledge and experience about acquiring a custom software system from software developers that are independent of the FAA. This includes knowledge about acquisition activities such as procurement, contracting, performance evaluation, and providing for future support of the software system. Knowledge about the appraisal and acquisition of COTS software, and its use and integration in software systems is becoming increasing critical for acquisition engineers.
Prerequisite Knowledge	Completion of the following modules (or equivalent knowledge): • LM 1: Basic Software Engineering • LM 2: Computing Fundamentals • LM 3: Software Domains • LM 5: Software Management (Advance) • LM 6: Product Quality Control • LM 11: Software Requirements
Description	This module provides an in-depth coverage of issues associated with software acquisition. This module covers the concepts, methods, processes, procedures, and techniques associated with procurement, contracting, performance evaluation, software management, and software quality control.
Module Objectives	Upon completion of this module the engineer will be able: • identify and discuss the different phases of software procurement and

	acquisition life cycle • understand the different acquisition strategies • familiar with the different negotiation techniques • understand different software standards, and use them to follow the progress of contractors • develop a plan and schedule that need to be met by the contractor, and assess the contractor performance based on them. • use different quality control techniques to assess the quality of the product
Module Content	The module will include the following topic areas: • Acquisition Life Cycle • System Life Cycle • Software Life Cycle • Procurement Process • Acquisition Strategies ➢ Competitive acquisition ➢ Two phase acquisition ➢ Sole-source acquisition ➢ Arts and techniques of negotiation ➢ Cost and price analysis • Software Management ➢ Establishing requirements ➢ Planning ➢ Schedule and cost control ➢ Standards identification • Performance management and assessment techniques ➢ Validation and verification ➢ Quality Assurance ➢ Review, inspection and test ➢ Metrics
Recommended Module Format and Learning Activities	• four hours of reading/study preparation • three days workshop • daily format ➢ Morning – lectures and discussions ➢ Afternoon – individual and group exercises • exercises ➢ Number of acquisition case studies and role playing ➢ Planning and tracking
Required Effort	28 hours
Additional Training	• 3 months of supplemental experiential learning involving software acquisition activities
Assessment	• Pre and post workshop self-assessment • Workshop exercise results
Resources	• Primary Resources: LR 73 • Secondary Resources: LR 69, LR 70, LR 71, LR 72, LR 74
References	• Marcinial, .J, and Raifer, D.J., *Software Acquisition Management : Managing the Acquisition of Custom Software Systems*, Wiley, 1990. • Vallabhaneni , S.R., *Auditing Purchased Software : Acquisition, Adaptation, and Installation*. • Glaseman, S., *Comparative Studies in Software Acquisition : Management Organization Versus the Development Process*. • Peter A. Kind, Jack Ferguson, "*The Software Acquisition Capability Maturity Model*", Software Engineering Institute, March 1997. • http://www.stsc.hill.af.mil/CrossTalk/1997/mar/acq_cmm.asp

IV. REFERENCES

1 Hilburn, T.B., Hirmanpour, I., Khajenoori, S., Qasem, A., *FAA-ARA Software Engineering Competency Study: Final Report*, December 1998.

2 Hilburn, T.B., Hirmanpour, I., Khajenoori, S., Qasem, A., Turner, R., *A Software Engineering Body of Knowledge, Version 1.0*, CMU/SEI-99-TR-004, Software Engineering Institute, Carnegie Mellon University, April 1999. http://www.sei.cmu.edu/publications/documents/99.reports/99tr004/99tr004abstract.html

3 Ibrahim, L., et. al*., The Federal Aviation Administration Integrated Capability Maturity Model*, Version 1.0, November 1997.

4 FAA, *Software Training Proposal: Training for Aircraft Safety engineers involved in approving software based systems*, Revision 0, June 19.,1998.

5 *ARA Intellectual Capital Investment Plan*, Office of Associate Administrator for Research and Acquisitions, FAA, 1997.

6 *ARA Role Workbook*, Office of Associate Administrator for Research and Acquisitions, FAA, 1998.

7 ARA Curriculum Group, *FY 1999 Goal, Outcomes, and Deliverables*, 1999.

8 FAA Contract, Award No.: DTFA0199P12250, Statement of Work, Phase II, Software Engineering Competency Study, 1999.

APPENDIX A: LEARNING RESOURCES

LR 1

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3) • Software Product Engineering (2)		

Type of Instruction:	Seminar
Name of the Institution:	ICE - Integrated Computer Engineering, Inc., Computers & Concepts Associates Division
Name of the Course:	Software Engineering Management
Reference:	http://www.candca.com/training.html
Location:	All courses are designed as "on-site" training workshops; presented at the client's facility or local hotel conference room. Prices are based on a minimum of 20 seats. Federal and Department of Defense customers qualify for our lower GSA/FEDSIM rates. Training funds can be placed on the GSA/FEDSIM contract and remain there until the customer charges against it for desired training or other project management services.
Duration:	5 days
Prerequisites:	None
Attendees:	In general, anyone involved in software acquisition or development will benefit from these courses and workshops. The Management Overview workshops are intended primarily for executives, senior managers and managers who are not software specialists.
Description:	**Course Objectives** Software Engineering Management provides system developers with the tools appropriate to the development and maintenance of software engineering applications including programming languages, language bindings and object code linking, and Computer Aided Software Engineering (CASE) environments and tools. This course details the application of a systematic, disciplined, and quantifiable approach to the development, operation and maintenance of software. Cultural aspects of software engineering are covered and are used as the basis for project definition. **Course Outline:** Module 1. Course Introduction and Overview Module 2. The Context of the Software Project: System Engineering Module 3. Overview of the Software Engineering Process Module 4. Planning the Work Module 5. Software Quality Assurance Management Module 6. Requirements Engineering and Analysis Module 7. Software Principal Best Practices Overview Module 8. Principal Best Practices #1: Formal Risk Management Module 9. Principal Best Practice #2: Agreement on Interfaces Module 10. Principle Best Practice #3—Formal Inspections Module 11. Principle Best Practice #4: Metrics Based Scheduling and Management Module 12. Principle Best Practice #5: Binary Quality Gates At The Inch Pebble

	Level Module 13. Principle Best Practice #6: Program Wide Visibility of Progress vs. Plan Module 14. Principle Best Practice #7: Defect Tracking Against Quality Targets Module 15. Principle Best Practice #8: Configuration Management Module 16. Principle Best Practice #9: People-Aware Management Module 17. SEI Assessments Module 18. Project Assessment and Redirection: Metrics, Assessments and Managing People Module 19. Conclusion and Summary Module 20. Appendix of “Buzzwords”

LR 2

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)		

Type of Instruction:	Seminar
Name of the Institution:	ICE - Integrated Computer Engineering, Inc., Computers & Concepts Associates Division
Name of the Course:	FAA Software Fundamentals Course
Reference:	http://www.candca.com/training.html
Location:	All courses are designed as "on-site" training workshops; presented at the client's facility or local hotel conference room. Prices are based on a minimum of 20 seats. Federal and Department of Defense customers qualify for our lower GSA/FEDSIM rates. Training funds can be placed on the GSA/FEDSIM contract and remain there until the customer charges against it for desired training or other project management services.
Duration:	5 days (commercial version also available)
Prerequisites:	None
Attendees:	In general, anyone involved in software acquisition or development will benefit from these courses and workshops. The Management Overview workshops are intended primarily for executives, senior managers and managers who are not software specialists.
Description:	**Course Objectives** The *FAA Software Fundamentals Course* teaches the FAA Aerospace Engineers (ASE) and Aviation Safety Inspectors (ASI) basic software engineering principles in order for the ASEs and ASIs to carry out their software-related functional responsibilities. The Engineers and Inspectors must be able to: a. Define software terms and be able to describe their use in relationship to the Applicant's software products and services. b. Identify the Applicant's activities in both the systems and software life cycles and their interrelationship, with emphasis on the software development process. c. Describe RTCA/DO-178B guidance as the basis for carrying out FAA functional responsibilities during the software life cycle. **Course Outline:** – An overview of the system life cycle. – The software life cycle. The software development process.

LR 3

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)		

Type of Instruction:	Seminar
Name of the Institution:	Lattice Limited
Name of the Course:	Introduction to Software Engineering
Reference:	http://www.lattice.co.uk/training/index.html
Location:	We typically deliver our courses at customer sites throughout the UK and Europe. We can also arrange courses for customers here in the UK, in London or in Cambridge. Lattice is also an experienced provider of tool training. Typically, this service is offered to tool vendors to provide their prospective users with comprehensive training programs correctly connected to the underlying software engineering. End-user companies may also benefit from existing tool courses, or in the development and delivery of specific tool courses.
Duration:	3 days
Prerequisites:	None
Attendees:	This course is appropriate for anyone who has some involvement in software engineering, is interested in raising the quality of software and who suspects that there are gaps in their knowledge. It will be helpful if participants have read Frederick Brooks' "Mythical Man-Month" and "No Silver Bullet".
Description:	This course aims to bring to the same level of awareness, the diverse range of people who find themselves developing large and complex software systems, and to cover some aspects of software engineering which are often not covered during typical software engineering tertiary education. Aims: – To survey the techniques and methods that can bring a significant improvement to the development of complex software systems – To bring everyone to a point where they are aware of most of the important discoveries in software development from the past fifteen years – To get people to a position where they know what techniques are available, the effects they will have and any problems that should be anticipated in their use, and where to go next to take particular techniques to the appropriate depth – To alert people to any important areas of software engineering that they didn't know they were unaware of

LR 4

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)		

Type of Instruction:	3.0 Credit Hours Semester Course
Name of the Institution:	National Technological University
Name of the Course:	SE 510-C - Software Engineering I
Reference:	http://www.ntu.edu/2/secrs.htm For Academic Questions Contact: Phil Barry (612) 624-8311 FAX (612) 625-0572 E-mail: barry@cs.umn.edu For Administrative Questions Contact: Fran Schirmers (612) 624-2332 FAX (612) 626-0761 E-mail: unite@cs.umn.edu
Location:	University of Minnesota
Duration:	30 (75 minute) lectures plus final
Prerequisites:	Programming languages and experience of developing 1000-line programs.
Attendees:	designers/managers
Description:	Advanced introduction to software engineering. Software life cycle; development models; software requirements analysis; software design, coding, and maintenance. Course Outline by Topical Areas: – Software Life Cycle and Development Models, Waterfall, spiral, prototyping, reusability. Best practices and worst practices, Process modeling and management. – Software Requirement Analysis, Object-oriented analysis, Analysis activities, Evaluation criteria, Survey of analysis techniques, – Software Design, Design architecture, Object-oriented design, Design patterns, Design rules. – Software Coding, Coding rules Software Maintenance, Program slicing, Ripple effect analysis, Data-centered program understanding, Business rule extraction

LR 5

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)		

Type of Instruction:	Seminar
Name of the Institution:	National Technological University
Name of the Course:	Principles of Software Requirements Engineering
Reference:	http://www.ntu.edu/1/atmp/1999Courses/mc99050402.htm
Location:	One live
Duration:	3-hour broadcast
Prerequisites:	Software development or management familiarity
Attendees:	This course is intended for software managers, project leaders, programmers and software engineers who are interested in structured and object- oriented approaches to software development. For those already familiar with basics of requirements engineering, this course will serve as a refresher. For those with little or no knowledge of requirements engineering, this course will provide a starting point, familiarizing you with concepts, methods, and approaches to requirements analysis and specification.
Description:	Large-scale software systems cannot successfully be developed without a complete understanding and representation of the system's requirements. Requirements engineering helps discovery, refinement, modeling and specification of all functional and non-functional elements of a system. A complete analysis/specification leads to high-quality software design and implementation, more reliable and maintainable software components and a reduction of development and maintenance costs of the software system over its life cycle. In this 3-hour course, we will introduce basic concepts and principles of software requirements engineering, its tools and techniques, and familiarize participants with the most common methods for modeling software systems. Both function-based and object-based approaches to system analysis and specifications will be discussed.

LR 6

Category of knowledge	Area of knowledge	Unit of knowledge
• Computing Fundamentals (1)	• Computer Architecture (1.2)	

Type of Instruction:	E-book
Name of the Institution:	Computational Science Education Project
Name of the Course:	Computer Architecture
Reference:	http://csep1.phy.ornl.gov/CSEP/CA/CA.html
Location:	On line
Duration:	
Prerequisites:	
Attendees:	
Description:	– Overview – Basic Computer Architecture – High Performance Computer Architecture – Exercises References

LR 7

Category of knowledge	Area of knowledge	Unit of knowledge
• Computing Fundamentals (1)	• Programming Language (1.5)	• Programming Paradigms (1.5.2)

Type of Instruction:	Course
Name of the Institution:	National Capital Training Center
Name of the Course:	BCOMP719 Structured Programming Techniques
Reference:	http://grad.usda.gov/cat/page30.html
Location:	600 Maryland Avenue SW, Suite 280 Washington, DC 20024-2520 Phone: (202) 314-3400 FAX: (202) 479-6810 TDD: (202) 314-3450 TOLL-FREE: (888) 744-GRAD
Duration:	3-day
Prerequisites:	
Attendees:	Participants should have taken Basic Concepts of Data Processing or have equivalent work experience.
Description:	Programming regardless of the language requires a special kind of logical thought process, and a knowledge of programming terminology. This three-day course is designed to provide a basis for specific programming language courses. It emphasizes problem definition and description, program definition, logic and flowcharting. It will review systems software, survey job control language, and examine the use of symbolic and problem-oriented language, and program documentation. Objectives: Upon completion of this course participants will: – develop the skills necessary to state with precision the steps to follow in problem solving – learn to apply the rules and requirements of structured programming – become familiar with basic concepts and terminology of programming (loops, assignment statements, identifiers, etc.) – learn to use structured flowcharts, pseudo codes and structured design charts – become aware of various programming languages currently available and their applications – be able to communicate effectively with programmers

LR 8

Category of knowledge	Area of knowledge	Unit of knowledge
• Computing Fundamentals (1)	• Programming Language (1.5)	• Programming Paradigms (1.5.2)

Type of Instruction:	Course
Name of the Institution:	Graduate School, USDA. The Government's Trainer
Name of the Course:	BCOMP968 Overview of Object-Oriented Technology
Reference:	http://grad.usda.gov/cat/page54b.html
Location:	Denver 12345 W. Almeda Pkwy Suite 303 Lakewood, CO 80228 (800) 787-9074 / (303) 969-5807 Fax (202) 479-4975 e-mail: appliedtech@grad.usda.gov
Duration:	1-day course
Prerequisites:	
Attendees:	Computer systems analysts, programmers, managers, and individuals who are interested in understanding OO concepts and techniques and how they may work in their organization.
Description:	Object oriented technology (OT) is an integrating technology for client server, complex systems, CASE, multimedia, distributed computing, work flow computing, web and other information technology areas. Object technology applies to all phases of software development including requirements, analysis, design, and implementation as well as enterprise analysis, information engineering and business reengineering. OT represents a major paradigm shift from traditional system development approaches and emphasizes the encapsulation of both the data and behavior aspects of objects in the system. This overview course covers the major concepts and components of object technology in the context of the SDLC (system/software development life cycle.) It also covers the role of standards and, in particular, the role of the new standard Unified Modeling Language (UML). The course also presents architectures and the use of pattens in represent models of systems. In addition, the course covers implementation issues for OO programming languages and object date management systems (ODMSs) and their relationships to older traditional methods. Management issues regarding reuse, the transitioning to object technology and object project management are discussed throughout the course.

LR 9

Category of knowledge	Area of knowledge	Unit of knowledge
• Computing Fundamentals (1)	• Computer Architecture (1.2) • Operating Systems (1.4)	

Type of Instruction:	Streaming Video Course
Name of the Institution:	College of Engineering University of Texas at Arlington, PO Box 19019, Arlington, TX 76019
Name of the Course:	CSE 3322 Computer Architecture I
Reference:	http://engineering.uta.edu/html/vidcourses.html
Location:	On line
Duration:	
Prerequisites:	Digital logic design, basics of: operating systems, computer organization, processor architecture, hardwired, and micro-programmed control unit.
Attendees:	
Description:	Hardware and software structures found in modern digital computers. Topics include instruction set architecture, processor architecture, memory architecture, input/output architecture, inter-connection schemes, and memory management. Emphasis is placed on the hardware and software interfaces within a computer system.

LR 10

Category of knowledge	Area of knowledge	Unit of knowledge
• Computing Fundamentals (1)	• Mathematical Foundations (1.3)	• Discrete Mathematical Structures (1.3.2)

Type of Instruction:	Streaming Video Course
Name of the Institution:	College of Engineering University of Texas at Arlington, PO Box 19019, Arlington, TX 76019
Name of the Course:	CSE 2315 Discrete Structures
Reference:	http://engineering.uta.edu/html/cse2315.html
Location:	On line
Duration:	
Prerequisites:	CSE 1320 and Calculus I; programming skills in Pascal and/or C.
Attendees:	
Description:	This course augments the student's theoretical foundation of Computer Science in the subject areas of formal logic, mathematical proof techniques, sets, combinatorics, functions and relations, Boolean algebra, graphs, and graph algorithms.

LR 11

Category of knowledge	Area of knowledge	Unit of knowledge
• Computing Fundamentals (1)	• Algorithms and Data Structures (1.1)	• Design of Algorithms (1.1.2) • Analysis of Algorithms (1.1.3)

Type of Instruction:	Streaming Video Course
Name of the Institution:	College of Engineering University of Texas at Arlington, PO Box 19019, Arlington, TX 76019
Name of the Course:	CSE 5311Design and Analysis of Algorithms
Reference:	http://engineering.uta.edu/html/cse2315.html
Location:	On line
Duration:	
Prerequisites:	Algorithms and Data Structures (CSE 2320) and Theoretical Concepts in Computer Science and Engineering (CSE 3315).
Attendees:	
Description:	Techniques for analyzing upper bounds for algorithms and lower bounds for problems. Problem areas include: sorting, data structures, graphs, dynamic programming, combinatorial algorithms, organization of numerical computations, introduction to parallel models.

LR 12

Category of knowledge	Area of knowledge	Unit of knowledge
• Computing Fundamentals (1)	• Algorithms and Data Structures (1.1)	

Type of Instruction:	Seminar
Name of the Institution:	Advanced Strategies, Inc.
Name of the Course:	Data Modeling - Practitioners
Reference:	http://www.advancedstrategiesinc.com/courses/c12.htm
Location:	3980 Dekalb Technology Parkway in Atlanta, Ga.
Duration:	
Prerequisites:	Business Object Modeling
Attendees:	This course is targeted for database designers and individuals responsible for designing/coding application software to access data.
Description:	This course presents the evolution from the object model (E/R diagram) to the physical data model (relational tables). Three models are used to manage this complex process: the conceptual data model, the logical data model, and the physical data model. The constraints at each stage and the design trade- offs involved are presented. The conceptual and logical data models are documented using the data structure diagram. The student shall gain the skills necessary to construct high quality data structure designs in his/her actual work setting. Course Outline: – Data Design in the System Development Life Cycle – Data Structure Diagramming – Logical Record Linkage and Dependency – Database Navigation – Performance Optimization vs. Flexibility – Data Structure Adjustment – Transaction Analysis SM – Implementation Data Dependencies: – DBMS Data Dependencies: Note: In special situations, this course can be segmented into modules as follows: Data Modeling Overview, Solid, and Advanced Classes. Please talk with your Advanced Strategies Education Advisor for details.

LR 13

Category of knowledge	Area of knowledge	Unit of knowledge
• Computing Fundamentals (1)	• Algorithms and Data Structures (1.1)	

Type of Instruction:	Seminar
Name of the Institution:	Advanced Strategies, Inc.
Name of the Course:	Data Modeling - Solid
Reference:	http://www.advancedstrategiesinc.com/courses/c13.htm
Location:	3980 Dekalb Technology Parkway in Atlanta, Ga.
Duration:	Four Days
Prerequisites:	Object Modeling Solid
Attendees:	Database Designers, Data Architects, and Analysts
Description:	At the end of this workshop, participants will gain the skills necessary, independently, to construct high quality logical data specifications using data structure diagrams in their actual work settings. Specifically, the student will be able to take the object models produced in Business Object Modeling Basic and Solid courses, transforming them into logical data models which support physical relational database design. Course Outline: – Data Design in the System Development Life Cycle – Database Navigation – Performance Optimization vs. Flexibility – Data Structure Adjustment – Transaction Analysis Awareness of Implementation Dependencies: – Current Business Policies – Planned Data Redundancy – Derivable Data – Security and Audit Requirements Awareness of DBMS Data Dependencies: – Indexing Decisions – Backup, Recovery, Journaling – The Relational Model Awareness of Normalization

LR 14

Category of knowledge	Area of knowledge	Unit of knowledge
• Computing Fundamentals (1)	• Computer Architecture (1.2)	

Type of Instruction:	Workshop
Name of the Institution:	ISCA 2000 Sponsored by ACM SIGARCH and IEEE Computer Society TCCA
Name of the Course:	The 27th Annual International Symposium on COMPUTER ARCHITECTURE
Reference:	http://www.cs.rochester.edu/~ISCA2k
Location:	Vancouver, British Columbia, Canada June 12-14, 2000
Duration:	2 days
Prerequisites:	
Attendees:	
Description:	– Innovative microarchitecture and implementation techniques – Multiprocessors, multicomputers, and distributed architectures – Novel architectures and computing techniques – Architectural implications of application characteristics – Application-specific or special purpose architectures – Performance evaluation and measurement of real systems – Memory hierarchy and I/O system architecture – Impact of technology on architecture

LR 15

Category of knowledge	Area of knowledge	Unit of knowledge
• Operating Systems and Networks (1)	• Computer Architecture (1.2)	• Communications and Networks (1.2.4)

Type of Instruction:	Seminar
Name of the Institution:	TNN – The Network Network
Name of the Course:	Understanding Data Communications
Reference:	http://www.thenetworknetwork.com/training/
Location:	Many cities available: go to the site to find the schedule (Irvine, Chicago, San Francisco, Sydney, Wellington, Auckland)
Duration:	2 days
Prerequisites:	
Attendees:	– Computer and communications personnel who have had little formal exposure to data communications and wish to be brought rapidly up-to-speed – Technical staff starting out in their information networking education program – Management personnel who need an understanding of networking and internetworking in order to guide developments within their own organization – Experienced data communications personnel who wish to update and round out their knowledge and gain new insights into current and emerging methodologies – All non-technical personnel who need to cut through the fog of business networking
Description:	Understand the fundamental concepts and principles of data communications and Networking Essentials. Including: – Network Components / Real-World Networks / Building a LAN / Configuring The Server / Configuring the Client / Administering Your Network / Remote Access / – Wide Area Networks / Troubleshooting and Managememt – Understand the major features of the OSI 7-layer model and defacto standards such as SNA, TCP/IP, DECnet, IPX, and how the two can be integrated to maintain flexibility for the future – Gain a good understanding of currently available data communications services, both national and international. – Find out the importance, and the pros and cons, of various cabling systems, including coax, twisted pair, fiber optic, wireless, and the flexibility offered by Dedicated Switched Ethernet – Learn all about the major protocols (Ethernet, Token Ring, FDDI, TCP/IP, MANs, WANs) and how they can be linked to form an integrated network Internet, Intranet , and Extranet design guidelines – Applications and business solutions – Learn all about future developments and directions. Find out which technologies are likely to become obsolete before you invest in them

LR 16

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Domains (4)	• Real-Time Systems (4.6)	

Type of Instruction:	Distance Learning
Name of the Institution:	Carnegie Mellon University
Name of the Course:	Introduction to Real-Time Software Systems
Reference:	http://www.distance.cmu.edu/info/courses/real.html
Location:	
Duration:	
Prerequisites:	– Proficiency in at least one high-level programming language used to develop real-time – software (e.g., C, C++, or Ada). – Proficiency in a software design notation. – Knowledge of operating system concepts taught in an undergraduate operating system course. A student may acquire the prerequisite knowledge via an undergraduate course, on-the-job training, or an independent study.
Attendees:	
Description:	The primary purpose of this course is to present an overview of real-time software engineering. The course focuses on basic concepts, terminology, and problems of real-time applications. You will learn about constraints which distinguish real-time applications from other applications. Though the course focuses on software solutions to real-time problems, there will be a discussion of hardware components commonly used in real-time computing systems as well as hardware/software interfaces. Software issues which will be addressed include specification of system/software requirements and design, scheduling, software architectures for real-time software systems, languages and operating systems for real-time computing, and real-time problems in a distributed processing system.

LR 17

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Domains (4)	• Real-Time Systems (4.6)	

Type of Instruction:	Seminar
Name of the Institution:	Software Arts & Sciences
Name of the Course:	Schedulability Analysis for Hard Real-Time Systems
Reference:	http://www.classwide.com/training/schedula.htm Box 891591 Houston, Texas, USA 77289-1591 Telephone (281)648-3165 FAX (281)648-3165 General Information: info@classwide.com
Location:	On site
Duration:	1 day minimum, 2 days with POSIX or Ada 95 support material and laboratory exercises
Prerequisites:	No programming experience is required. However, professional programmers with previous real-time experience in a high-level language will benefit most from this seminar.
Attendees:	
Description:	This detailed, language-independent seminar explores the approaches available for guaranteeing deadlines in hard real-time systems. The seminar begins with an examination of the traditional cyclic executive, providing the rationale for modern process-oriented approaches. The foundation for process-oriented designs is then provided, including simple priority-based preemptive scheduling and why it is inadequate alone. Upon this foundation, the concept of deadline scheduling is introduced as a deterministic scheduling approach able to guarantee deadlines will be met. The techniques of deadline scheduling are then presented in detail, beginning with scheduling purely periodic process sets. Emphasis is given to schedulability analysis such that deadline viability can be determined prior to execution. As such, Rate Monotonic Analysis (RMA) is covered extensively, as are the more recent advances for analyzing schedulability than the RMA Utilization test. Discussion of the complexities introduced by including aperiodic and sporadic (event-driven) processes leads to the use of Deadline Monotonic Analysis as an alternative to RMA. Finally, processes that block for synchronization and communication – representing realistic applications – are added to the schedulable process set. The additional techniques for schedulability analysis are thus explored, including priority inheritance as a means of dealing with unbounded priority inversions, and protocols to increase the effectiveness of priority inheritance. The presentation closes with practical suggestions for further reading. Laboratory work is included to reinforce the material presented. Outline 1.Introduction

	1.Requirement for Real-Time Systems 2.Hard Deadlines 3.Soft Deadlines 2.Cyclic Executives 1.Advantages 2.Risks 3.Process-Based Designs 1.Advantages and Risks 2.Periodic, Aperiodic and Sporadic Processes 3.Priority-Based Schedulers 4.Preference Scheduling 1.Semantic Importance 2.Inadequacies 5.Deadline Scheduling 1.Deadline Mapping 2.Schedulability Analysis 3.Optimal Scheduling Schemes 4.Schedulable Processes Sets 6.Scheduling Periodic Processes 1.Rate Monotonic Analysis 2.Analysis via Timelines 3.Utilization Test 4.Response Time Test 5.Transient Overloads 7.Including Aperiodic & Sporadic Processes 1.Bandwidth-Preserving Algorithms 2.Deadline Monotonic Analysis 8.Including Blocking Processes 1.Blocking 2.Priority Inversion 3.Priority Inheritance 4.Ceiling Priority Protocol 5.Original Ceiling Priority Protocol 6.Immediate Ceiling Priority Protocol 9.Including System Overheads 1.Context Switches 2.Interrupt-Driven Sporadic Processes 3.Real-Time Clock Interrupt Handling 10.Case Study 11.Concluding Remarks 12.Recommended Reading

LR 18

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Domains (4)	• Human-Computer Interaction (4.3)	

Type of Instruction:	NTU Semester Credit Hours: 3
Name of the Institution:	National Technological University
Name of the Course:	SE 735-N User Interface Design
Reference:	http://www.ntu.edu/1/credit/se735n.htm
Location:	Department of Computer Science and Engineering Southern Methodist University P.O. Box 750122 Dallas, TX 75275 (214) 768-3080 (972) 497-4033 FAX (214) 768-3085 E-mail: mdiaz@rsn.hp.com
Duration:	15 (180 minute) lectures
Prerequisites:	None. Programming is not required. Some experience using a graphical user interface application (e.g. Windows, Macintosh, X Windows) would be useful.
Attendees:	
Description:	The course will discuss both the interface itself and, more importantly, the process by which one is developed. Though a brief overview is presented, the course is not about the psychology of human factor aspects of interface design. Rather, the course is geared towards the software-engineering aspects of development. **Course Outline**: – User Interface Guidelines – Interface Styles and Widgets – Human Factors Considerations – Iterative Development – Systems Analysis – Quantifying Usability – User Interface Representation – Rapid Prototyping/Visual Basic – Usability Testing

LR 19

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Domains (4)	• Human-Computer Interaction (4.3)	

Type of Instruction:	Seminar
Name of the Institution:	National Technological University
Name of the Course:	Web Development with HTML Course Code: MC99062401
Reference:	http://www.ntu.edu/1/atmp/1999Courses/mc99062401.htm
Location:	One live
Duration:	6-hour broadcast
Prerequisites:	A working knowledge of your operating system and experience with browsing the World Wide Web
Attendees:	Anyone interested in creating great-looking, platform-independent documents for the World Wide Web
Description:	HTML is the glue that holds together the massively-expanding phenomena known as the World Wide Web. At the core of all Web development is the knowledge of Hypertext Markup Language (HTML) and the proper style of its use. This course is a fundamental from-the-ground-up approach to understanding the core HTML 3.2 language, as well as an overview of some of the current "extension tags" that exist for certain browsers. Topics covered will include format- and context-oriented tags, formatting text, in-line images, tables, frames, forms, and more.

LR 20

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Domains (4)	• Real-Time Systems (4.6)	

Type of Instruction:	Seminar
Name of the Institution:	University Consortium for Continuing Education
Name of the Course:	Real Time Software Design
Reference:	http://www.ucce.edu/onsites/css.html
Location:	On site UCCE specializes in providing on-site courses in engineering and management in a technical environment. UCCE has a vast array of quality short courses available to be offered at your facility and may be scheduled at your convenience. For additional information including detailed brochures on individual courses or a proposal to offer a course at your facility, contact Tom Mincer for *On-Site Proposals & Information* or call (818) 995-6335.
Duration:	
Prerequisites:	
Attendees:	
Description:	Software specification and design techniques developed for traditional applications typically are inadequate when they are applied to real time systems. This five-day tutorial identifies the issues unique to real time systems, describes mathematical models needed to resolve these issues, and then describes a variety of real-time notations and techniques. The special role of requirements specification in traditional and emerging models of the software development life cycle what information should be present in a requirements specification the mathematical complexity of real-time applications the capabilities and limitations of traditional requirements specification techniques the mainline real-time specification methods

LR 21

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Domains (4)	• Human-Computer Interaction (4.3)	

Type of Instruction:	Computer Based Instruction
Name of the Institution:	National Program Office for Computer Based Instruction
Name of the Course:	GUI Design Fundamentals
Reference:	http://faawbt.jccbi.gov/cbtlib/htmlplan/cbtweb/curicula/courses/ADPG01E/ADPG01E.htm
Location:	On line
Duration:	4 hours
Prerequisites:	An understanding of basic windows and Web terminology; visual programming experience and HTML or previous Web design experience an advantage
Attendees:	Those involved with designing or implementing graphical user interfaces (GUIs), including software project managers, programmers, analysts and designers, software engineers, and technical writers
Description:	Course Aim: To teach the principles of effective GUI design for windows-based applications Topics Covered – GUI design principles – Usable GUIs – Design principles – Images, icons, and color – User requirements – GUI controls – Window design principles – Usability Learning Objectives – After taking this course, the student should be able to describe the principles of graphical user interface design – differentiate between effective and ineffective interfaces – list and describe key attributes of windows – develop effective navigation through the interface – know when to choose color, graphics, audio, and animation, and when not to

LR 22

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Project Management (3.1) • Software Risk Management (3.2) • Software Process Management (3.5)	

Type of Instruction:	Computer Based Instruction
Name of the Institution:	National Program Office for Computer Based Instruction
Name of the Course:	Project Management: Fundamentals
Reference:	http://faawbt.jccbi.gov/cbtlib/htmlplan/cbtweb/curicula/courses/PROJ01E/PROJ01E.htm
Location:	On line
Duration:	3 hours
Prerequisites:	A basic understanding of the need for project management and an interest in the principles of effective project management.
Attendees:	Students interested in the principles of project management; trainee project managers; consultants routinely engaged in project management; experienced project managers wishing to refresh their thinking on the principles of project management; program managers and senior managers employing or managing project managers
Description:	To provide an introduction to the principles of project management : Roles and responsibilities – Projects and programs – The project manager – Functional managers Project team members – Project variables – Scope – Scheduling activities – Risk – Quality – Resources Project processes – Planning – Controlling – Reporting – Concluding Information Store – Site-level Information Store configuration Public and private information stores –

LR 23

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Configuration Management (3.4)	

Type of Instruction:	Seminar
Name of the Institution:	Software Productivity Center
Name of the Course:	Software Configuration Management
Reference:	http://www.spc.ca/training/courses/index.htm
Location:	BCIT Downtown Campus, 555 Seymour St., Vancouver, BC
Duration:	
Prerequisites:	
Attendees:	Software engineers, team leaders, programmers and technically-oriented software product managers and information systems managers.
Description:	Course Overview Producing high-quality software is crucial. Vice presidents of software development firms have stated that the most significant effect in their process improvement activities has often been the start-up of a good configuration management (CM) organization and process. Participants will acquire a solid understanding of the CM methods considered essential to achieving high levels of software quality. They will learn how to implement these methods in their companies and projects. This course presents many tips and techniques from leading companies that view CM as the norm. Participants will learn how important CM is in terms of creating successful software, and in terms of compliance with the ISO 9000 standard. Participants will learn the CM practices required to control software products through their life cycle from initial creation through construction, to testing, to delivery and maintenance. Topics – software product management – configuration control principles – change request process and information – configuration identification – delta-storage concepts – organizing for configuration management – determining configuration management requirements – software configuration management plan – making configuration management happen

LR 24

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2) • Software Management (3)		

Type of Instruction:	Seminar
Name of the Institution:	DACS
Name of the Course:	System Engineering
Reference:	http://www.dacs.dtic.mil/training/sysed.shtml
Location:	ITT SystemsCorporation 2560 Huntington Ave. Alexandria, Virginia 22303 (703) 960-4906 On-Site options are also available. Call the DACS Customer Liaison for details.
Duration:	
Prerequisites:	
Attendees:	The course is designed for professionals involved in systems management and development either in a direct engineering role or oversight management role. It would be extremely useful as a familiarization course. Thus, the course is intended for those persons that have little knowledge of system engineering, to those that have knowledge in distinct areas or wish to update on recent events.
Description:	The seminar will provide an understanding of System Engineering as it is practiced in the DoD and associated government agencies. It will cover basic concepts of System Engineering to include: definitions, standards used, new developments in the area, and related topics in areas such as Corporate Information Management (CIM) and Business Process Reengineering (BPR). Exercises will be used to enhance the training. Attendees can expect to acquire enough insight into this topic to be able to apply the concepts contained herein. **Course Outline :** – System Engineering Overview – Requirements Engineering – Software Development – System Architecture – Risk Management – Performance Measurement and Evaluation – Life-Cycle Acquisition – Life-Cycle Costing – Support and Specialty Engineering – System Software Engineering Tools

LR 25

Category of knowledge	Area of knowledge	Unit of knowledge
Software Management (3)		

Type of Instruction:	Seminar
Name of the Institution:	DACS
Name of the Course:	Software Engineering for Program Managers
Reference:	http://www.dacs.dtic.mil/training/seng4pro.shtml
Location:	ITT SystemsCorporation 2560 Huntington Ave. Alexandria, Virginia 22303 (703) 960-4906 On-Site options are also available. Call the DACS Customer Liaison for details.
Duration:	
Prerequisites:	
Attendees:	The course is designed for professionals involved in software engineering or program management. One of the main course objectives is to improve understanding between program managers and software engineers therefor both will benefit from this course.
Description:	This course will highlight the following areas: – Program Management and Software Engineering – Software Process Maturity – Life Cycle Management – Software Project Management – Software Risk Management

LR 26

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)		

Type of Instruction:	Seminar
Name of the Institution:	Oregon Graduate Institute of Science and Technology Center for Professional Development
Name of the Course:	Estimating, Measuring, and Controlling Software Projects
Reference:	http://www.ogi.edu/CPD/courses/
Location:	20000 N.W. Walker Road Beaverton, Oregon 97006-8921 Fax: +1-503-748-1686.
Duration:	2 days
Prerequisites:	No special background is required. Those with more experience and knowledge of software engineering and software management benefit most from this course. It is recommended, but not required, that attendees complete the short course Planning and Estimating Software Projects before attending this course.
Attendees:	Project managers, software acquirers, software developers, quality engineers, systems engineers, division managers and all others who are concerned with developing satisfactory software products and software components within the constraints of schedule, budget and available resources will benefit from this course.
Description:	This two-day course presents methods, tools, and techniques for estimating effort, schedule, resources requirements, and risk factors as determined by required product features and quality attributes; techniques for measuring schedule progress, resource utilization, product features and quality attributes attained; process effectiveness; and risk indicators. In addition, methods for measuring and controlling each project factor and the interactions among project factors will be presented. Methods for estimating cost and schedule to complete an on-going project will be described, and techniques for making tradeoffs among schedule, budget, resources, product attributes, and risk factors will be presented. Course Outline 1. The nature of project management 2. Software project foundation elements 3. Risk management procedures 4. Selecting a software development model 5. Planning the big three: effort, schedule, and resources 6. Systematic estimation techniques 7. Planning the supporting processes 8. Planning for measurement and control

LR 27

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Quality Management (3.3)	• Software Metrics (3.3.3)

Type of Instruction:	Seminar
Name of the Institution:	DACS
Name of the Course:	Software Measurement: Implementation and Practice
Reference:	http://www.dacs.dtic.mil/training/soft.meas.shtml
Location:	ITT SystemsBuilding 2560 Huntington Avenue Alexandria, Virginia 22303 USA (703) 960-4906 The office is located one block from the Huntington Metro stop (yellow line); parking is available at the Kaman building. On-Site options are also available. Call the DACS Customer Liaison for details.
Duration:	?
Prerequisites:	
Attendees:	This course is designed for the software professional involved in project management, oversight for software intensive projects, software acquisition management, or software development and engineering who has experience with software and software development but is not familiar with measurement or measurement practice.
Description:	Software Measurement Implementation and Practice will highlight the following areas: Measurement and Metrics; Measurement Paradigms; and the Experience Factory. This seminar will provide a basic understanding of measurement methods and problems, discuss the practical aspects of metric collection, give examples of metrics and management indicators, discuss measurement initiatives underway throughout the world, develop an understanding of measurement techniques that are utilized in practice, discuss new paradigms for measurement, and explain the concept of an Experience Factory. Attendees will acquire solid knowledge of what measurement is, how to implement it, and how to use it in software engineering practice. In addition, the seminar will provide attendees with new techniques for implementing measurement in a process-based reuse environment. – History of Software Measurement – Metrics (McCabe's Cyclomatic Complexity) and Management Indicators – Data and Metric Collection Methods – Using Measurement in Software Engineering Practice – Measurement Paradigms Software Quality Framework, Goal / Question / Metric, Gilb's metrics, etc. – The Experience Factory – Data Repositories – Measurement Experiments

LR 28

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Risk Management (3.2)	• Risk Analysis (3.3.1)

Type of Instruction:	Course
Name of the Institution:	American Society for Quality
Name of the Course:	Risk Analysis Tools Techniques
Reference:	http://www.asq.org/products/courses/fall/engistat.html#eng10
Location:	Newark, NJ October 19-21, 1999 Course # 99235 ASQ's Customer Service Center: American Society for Quality, 611 East Wisconsin Avenue, P.O. Box 3005, Milwaukee, WI 53201-3005 Tel: 1- 800-248-1946 or 414-272-8575, fax 414-272-1734, e-mail: cs@asq.org
Duration:	3 Days
Prerequisites:	
Attendees:	Project team managers, design engineers, production engineers, development staff, quality assurance and regulatory affairs specialists, market researchers, and senior sales representatives.
Description:	Benefits: Know how to organize efforts to improve product safety and reliability. Get a road map for "before-the-event" activities that address risk. Appreciate how risk analysis can reduce start-up costs and time to market. Understand how risk analysis fosters teamwork and communication. Be ready to comply with regulatory (ISO, FDA, etc.) requirements. Highlights: – Know how to apply risk analysis as a product or process life cycle activity – Learn when and where to apply different risk analysis tools and techniques – Be able to apply three major risk analysis techniques: Preliminary hazard analysis, Fault tree analysis, Failure mode & effects analysis – Know how to document risk analysis activities as a living process – Understand organizational dynamics when conducting a risk analysis – Work with actual industry case studies

LR 29

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Quality Management (3.3) • Software Configuration Management (3.4)	• Software Quality Assurance (3.3.1)

Type of Instruction:	Seminar
Name of the Institution:	Computer Generated Solutions - Instructor Led Technical Training
Name of the Course:	Software Quality Assurance Techniques
Reference:	http://devel.netxactics.com/cgsinc/_training/ilt/q233.htm
Location:	Computer Generated Solutions, Inc. (World Headquarters) 1675 Broadway, New York, N.Y. 10019 Tel: (212) 408-3800 • Fax: (212) 977-7474
Duration:	4 Days
Prerequisites:	None
Attendees:	Anyone responsible for improving quality of software products
Description:	This course teaches the participant how to apply techniques known to improve software quality. As the process of developing computer programs has matured, an ever-widening range of applications has been produced. Consequently the user base has expanded, and user expectations have risen, with particular emphasis on improvement of program quality. Software requirements specifications, however, have usually lacked precise definitions of software quality, and many organizations do not identify specific software quality assurance responsibilities. – The software development process – Software quality assurance planning – Software configuration management – Quality measurement, analysis and corrective action – Reviews, audits and inspections – Tools, techniques and methodologies – Software development tools – Software development – Implementing a software quality improvement program

LR 30

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Product Management (3.1) • Software Quality Management (3.3) • Software Configuration Management (3.4)	• Software Quality Assurance (3.3.1) • Software Verification &Validation (3.3.2)

Type of Instruction:	Course
Name of the Institution:	American Society for Quality
Name of the Course:	Software Quality Engineering
Reference:	http://www.asq.org/products/courses/fall/softeng.html#soft4
Location:	Nashville, TN July 12-16, 1999, Course # 99189 Milwaukee, WI November 1-5, 1999, Course # 99247 ASQ's Customer Service Center: American Society for Quality, 611 East Wisconsin Avenue, P.O. Box 3005, Milwaukee, WI 53201-3005 Tel: 1- 800-248-1946 or 414-272-8575, fax 414-272-1734, e-mail: cs@asq.org
Duration:	4 Days
Prerequisites:	Knowledge of and/or work experience within the software quality assurance field is helpful.
Attendees:	Software quality specialists, software quality engineers, software process engineers, and quality engineers wishing to obtain a basic understanding of software quality engineering practices and principles.
Description:	This course explores the Body of Knowledge for ASQ's Software Quality Engineer certification program. Comprehensive overview of the skills and knowledge necessary to perform software quality engineering tasks: – Understand the software life cycle – Determine how to evaluate software quality activities and processes and determine whether or not they meet their intended purpose – Software TQM – Software life cycle – Project management – Software configuration management – Software audits, reviews, and inspections – Software metrics and other issues – Complimentary copy of ANSI/ISO/ASQC Q9000-3-1997

LR 31

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Quality Management (3.3)	• Software Metrics (3.3.3)

Type of Instruction:	Course
Name of the Institution:	American Society for Quality
Name of the Course:	Software Metrics
Reference:	http://www.asq.org/products/courses/fall/softeng.html#soft3
Location:	Milwaukee, WI August 10-11, 1999 Course # 99201 Boston, MA October 7-8, 1999 Course # 99184C ASQ's Customer Service Center: American Society for Quality, 611 East Wisconsin Avenue, P.O. Box 3005, Milwaukee, WI 53201-3005 Tel: 1- 800-248-1946 or 414-272-8575, fax 414-272-1734, e-mail: cs@asq.org
Duration:	2 Days
Prerequisites:	
Attendees:	Software quality engineers, software process engineers, software quality specialists, quality engineers
Description:	Benefits: – List the metrics Do's and Don'ts – Identify relevant industry initiatives – Define and implement a measurement program – Identify what to measure – Interpret and communicate the results – Create a corrective action path Highlights – Examine how a software metric is determined – Analyze actual case studies of software metrics in use

LR 32

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Project Management (3.1) • Software Quality Management (3.3)	

Type of Instruction:	Computer Based Instruction
Name of the Institution:	National Program Office for Computer Based Instruction
Name of the Course:	The Software Development Process: Management Practices
Reference:	http://faawbt.jccbi.gov/cbtlib/htmlplan/cbtweb/curicula/courses/MISG05E/MISG05E.htm
Location:	
Duration:	4 hours
Prerequisites:	Familiarity with a software process
Attendees:	Software process managers, software engineers, and team leaders; business managers with responsibility for a software process
Description:	To provide an overview of management practices, software quality, and software metrics Topics Covered: Project management principles – Strategic planning – The project plan – Managing technical people – Management of the software process Quality assurance – Process assessment and improvement – Testing and benchmarking – The Personal Software Process Software metrics – Measurement processes – Using software metrics – Data collection and analysis

LR 33

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Quality Management (3.3)	

Type of Instruction:	Seminar
Name of the Institution:	Software Quality Institute
Name of the Course:	Software Quality Assurance
Reference:	http://www.utexas.edu/coe/sqi/seminars/SoftwareQA.html
Location:	Pickle Research Campus University of Texas at Austin PRC MER Code:R9800 Austin, TX 78712-1080 Telephone: (512) 471-4874 or (800) 687-8012 Fax: (512) 471-4824 or send E-mail to info@sqi.utexas.edu.
Duration:	3 days
Prerequisites:	Management and development experience with software development projects
Attendees:	Key managers, leaders and software developers in an organization where software development or selection is a significant component of the business.
Description:	Providing software developers and managers at all levels the insight for making the software quality assurance processes visible and measurable, establishing the standards for planning, developing and monitoring those processes and instilling the commitment to do so is the objective of this course. Course Outline: – General concepts of quality - Quality work framework – Planning software quality assurance activities – Issues in quality measurement – Applying quality standards – Overview of the SEI Capability Maturity Model – Techniques for planning / defining requirements that lead to high quality – Defect management and prevention – Specific software quality technologies – SQA management practices – SQA processes for producing high quality software – Achieving the "ilities" of software (e.g., reliability, usability, maintainability, etc.) Roles, responsibilities and authority of the SQA group.

LR 34

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Configuration Management	

Type of Instruction:	Seminar
Name of the Institution:	Software Quality Institute
Name of the Course:	Software Configuration Management
Reference:	http://www.utexas.edu/coe/sqi/seminars/SoftwareQA.html
Location:	Pickle Research Campus University of Texas at Austin PRC MER Code:R9800 Austin, TX 78712-1080 Telephone: (512) 471-4874 or (800) 687-8012 Fax: (512) 471-4824 or send E-mail to info@sqi.utexas.edu.
Duration:	3 days
Prerequisites:	Experience with software project management or estimating.
Attendees:	Software project managers and technical contributors who are responsible for complete software engineering efforts.
Description:	To understand software configuration management as part of the software development process and to understand a range of techniques for doing software configuration management. Course Outline: – The configuration management process – Baselines – Identification of objects in the software configuration – Version control – Change control – Configuration audit – Status reporting

LR 35

Category of knowledge	Area of knowledge	Unit of knowledge
Software Management (3)		

Type of Instruction:	Seminar
Name of the Institution:	ICE - Integrated Computer Engineering, Inc., Computers & Concepts Associates Division
Name of the Course:	Advanced Software Management
Reference:	**http://www.candca.com/training.html**
Location:	All courses are designed as "on-site" training workshops; presented at the client's facility or local hotel conference room. Prices are based on a minimum of 20 seats. Federal and Department of Defense customers qualify for our lower GSA/FEDSIM rates. Training funds can be placed on the GSA/FEDSIM contract and remain there until the customer charges against it for desired training or other project management services.
Duration:	12 days
Prerequisites:	The benefit realized from training depends on prior knowledge and experience. Normally anyone who finds themselves in a position in which one or more of these courses and workshops would be useful will have the background necessary to benefit from the training. Lack of software-specific knowledge and experience may be offset in some cases by management and engineering experience in other fields. Non-technical managers should seriously consider taking one or more management overview workshops prior to taking other courses or workshops.
Attendees:	In general, anyone involved in software acquisition or development will benefit from these courses and workshops
Description:	*Advanced Software Management* is a comprehensive and intensive course which provides in-depth coverage of the military and commercial policies, standards, practices, methodologies and techniques necessary for the planning and management of large-scale software acquisition and development projects. The focus is on the elimination of cost and schedule overruns through the identification and management of program risks, and an emphasis on Software Best Practices. **Topics and Outline:** • Introduction and Background • Methods for Managers • Software Architectures: DII COE, JTA and ATA • System Safety: Analysis and Assurance • Security Considerations • Requirements: Information, Function, Interface and Integration • Configuration Management • Software Defects and Inspections • Project Planning, Estimation, Scheduling and Earned Value • Software Risk Management and Metrics • Metrics and Measures • Software Quality Management

	• Contracting for Software • Testing: Evaluating the Project • Independent Verification and Validation (IV&V) • Integrated Product Teams • People Management • The Software Risk Assessment Process • Summary, Conclusions and Recommendations

LR 36

Category of knowledge	Area of knowledge	Unit of knowledge
Software Management (3)		

Type of Instruction:	Distance Learning
Name of the Institution:	National Technological University
Name of the Course:	WS99091098 - Software Project Management Program
Reference:	http://www.ntu.edu/5/bytopic99.htm For Academic Questions Contact: Phil Barry (612) 624-8311 FAX (612) 625-0572 E-mail: barry@cs.umn.edu For Administrative Questions Contact: Fran Schirmers (612) 624-2332 FAX (612) 626-0761 E-mail: unite@cs.umn.edu
Location:	On line
Duration:	Thirty-one tape-delayed, 2-hour broadcasts
Prerequisites:	The series requires experience working with software engineering projects,primarily so the attendee can understand the problems and appreciate some of the practical difficulties. Basic college level mathematics is also required.
Attendees:	The target audience includes current and prospective software development leads and managers. It will also be of interest to system engineers, software process specialists, managers of disciplines related to software engineering (such as software configuration management), and program managers whose programs have a significant software component.
Description:	This program, designed to improve software management skills, will consist of approximately 29 short courses, organized into four series. Each series is designed to focus on a different aspect of software project management: software project planning, software project execution, software project measurement and analysis, and software productivity and quality engineering. This short course program is based on several courses from the SMU MS program in software engineering as well as the SEI Capability Maturity Model. Each short course will include one or more exercises that apply the principles to concrete examples typical of those found in the workplace. A certificate will be given to those students completing this series in its entirety, including turning in all individual exercises for evaluation. The "Software Project Planning" series will show how to plan and estimate a software development project. Using practical examples and proven techniques, the student will learn how to assess a software project; estimate software size, effort, cost and schedule; assess risks; and plan for a successful software development. The "Software Project Execution" series will follow on from the planning series, focusing on the execution phase of a software project. It emphasizes practical aspects of risk management, configuration management, quality engineering, and schedule management, as well as tracking and oversight. The "Software Project Measurement and Analysis" series will show how to define

	appropriate metrics and to make effective use of them without excessive cost or alienation of software engineering staff. The series will have two running themes: taking the proper measurements and using the measurements effectively. The first theme will begin with several principles of measurement theory, applying them for effective selection and analysis of metrics. The second will focus on understanding human behavior and using that knowledge for effective collection and application of metrics. Recommended metrics will be discussed throughout, with examples and exercises that help the participant apply the principles to their own application. The "Software Productivity and Quality Engineering" series will serve as a capstone to the program by addressing how to engineer quality into a software product while improving productivity and reducing cycle time. The instructor will show in practical terms how to apply a variety of techniques within the context of the software development process. Topics will include cost of quality and value-added analysis, software process improvement, software cycle time and productivity improvement, software reliability, and six-sigma techniques.

LR 37

Category of knowledge	Area of knowledge	Unit of knowledge
Software Management (3)		

Type of Instruction:	Workshops
Name of the Institution:	SEPO
Name of the Course:	Software Management for Executives Course
Reference:	http://sepo.nosc.mil/training.html If you are interested in this workshop, please contact the Software Engineering Process Office (SEPO) at (619) or DSN 553-6694 or send email to sepo@spawar.navy.mil
Location:	SEPO, D12 SPAWARSYSCEN 53560 HULL ST. SAN DIEGO, CA 92152-50001 PHONE: (619)553-6694 FAX: (619)553-6249 LOCATION: BUILDING 312, BARRACKS AREA
Duration:	8 hours
Prerequisites:	
Attendees:	All department heads, division managers, other upper-level executives, and sponsors who oversee other managers directly responsible for software-intensive projects. SEPO provides several software engineering training courses for SSC San Diego government employees. Under certain circumstances, on a space available basis, there are openings to government employees outside of SSC San Diego. For additional information on course schedules, registration requirements, and other questions not covered in the flyers below, please contact SEPO at (619) 553-6694 or email Elizabeth Gramoy, SEPO Director
Description:	An eight-hour workshop for SSC SD managers at the department, division, and branch levels on the fundamentals of fostering continuous improvement of software engineering and project management practices at the Center. This workshop covers how a process discipline provides the critical foundation for software project success, and how the Center is approaching the improvement of its software processes. This will be accomplished by discussing the rationale for the development and use of software engineering and management processes. The importance and application of measurement in tracking project progress and process improvement at various management levels also will be emphasized.

LR 38

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Project Management (3.1)	

Type of Instruction:	Seminar
Name of the Institution:	Oregon Graduate Institute of Science and Technology Center for Professional Development
Name of the Course:	Software Project Management Planning
Reference:	http://www.ogi.edu/CPD/courses/
Location:	20000 N.W. Walker Road Beaverton, Oregon 97006-8921 Fax: +1-503-748-1686.
Duration:	2 days
Prerequisites:	No special background is required. Those with more experience and knowledge of software and software management will benefit most from this course.
Attendees:	Project managers, software acquirers, software developers, systems engineers, division managers and all others who prepare project plans, approve them, or live with the consequences of planning will benefit from this course.
Description:	This two-day course presents the details of these topics and provides examples of methods, tools, and techniques that can be used to prepare software project management plans, to make estimates of effort and schedule, and to update estimates and plans as conditions change. In addition, a procedure for developing generic plans and tailoring them for individual projects will be presented. Course Objectives After completing this course, you will understand what elements to include in a software project plan, techniques for preparing those elements, and how to update your project plans as conditions change. In addition, you will understand how to develop a generic plan for your organization and how to tailor it for individual projects.

LR 39

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2) • Software Management (3)	• Software Testing (2.4) • Software Quality Management (3.3)	• Unit Testing (2.4.1) • System Testing (2.4.3) • Software Quality Assurance (3.3.1) • Software Metrics (3.3.3)

Type of Instruction:	Seminar
Name of the Institution:	Rice Consulting Services, Inc.
Name of the Course:	Introduction to Quality Assurance and Testing
Reference:	http://www.riceconsulting.com/
Location:	
Duration:	2 days (can be customized to a 1 day version)
Prerequisites:	None
Attendees:	This course is designed for users, testers, developers and managers who want to learn how to assure the quality of the software they deliver. Any developer, tester, user, or manager who wants to learn about the basics of software quality assurance and testing.
Description:	The course lays a foundation in the principles of quality and quality assurance. Then, techniques are presented that can make QA an effective force in your organization. The course concludes with developing your own action plan for quality. **Topics:** – Concepts of Quality: Lessons From the Gurus of Quality – Concepts of QA – Software Risks – Software Testing Basics – The Purpose of Testing – The Testing Organization: Roles and Responsibilities – How to Develop Quality Requirements – Software Testing Methods (Unit Testing, Path Testing, Test Planning, Regression Testing, System Testing, Stress Testing, Automated Testing, Path Testing) – Measurements and Metrics - How to Decide What to Measure - Presentation Ideas – Developing an Action Plan for Quality for Your Organization

LR 40

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Quality Management (3.3)	• Verification & Validation (3.3.2)

Type of Instruction:	Seminar
Name of the Institution:	Richard Ball & Associates Inc.
Name of the Course:	Software Inspections and Walkthroughs
Reference:	http://www3.pei.sympatico.ca/ball/index.html
Location:	
Duration:	2 days
Prerequisites:	None
Attendees:	– IT practicing professional – Quality Assurance staff, including Inspection Moderators – Users who will be evaluating software deliverables
Description:	Customers are expecting ever higher standards of software quality. At the same time, the complexity of software is increasing, and it is increasingly recognized that machine-based testing, by itself, is not a cost-effective technique for ensuring the quality of software. In order to meet these demands, organizations are increasingly turning towards software inspections to ensure the cost-effective production of quality software. This in-depth seminar covers software inspections, walkthroughs, and other forms of human-based reviews. The seminar is suitable for organizations thinking about introducing inspections, or needing to improve existing inspection processes. Often, inspections stall or fail because they are introduced without proper training. This seminar helps overcome that deficiency. **Course Outline** Software Quality Issues Types of Technical Reviews Walkthroughs Baseline Inspection Characteristics Inspection Roles The Inspection Process Ten Keys to Inspection Success Inspections Across the System Development Lifecycle Administration of Inspections What Data You Should Collect Implementing Inspections Root Cause Analysis

LR 41

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Quality Management (3.3)	• Verification &Validation (3.3.2)

Type of Instruction:	Seminar
Name of the Institution:	Software Quality Institute
Name of the Course:	Verification and Validation Processes and Methods
Reference:	http://www.utexas.edu/coe/sqi/seminars/table.html
Location:	Pickle Research Campus University of Texas at Austin PRC MER Code:R9800 Austin, TX 78712-1080 Telephone: (512) 471-4874 or (800) 687-8012 Fax: (512) 471-4824 or send E-mail to info@sqi.utexas.edu.
Duration:	2 days
Prerequisites:	Experience with software project management or life cycle support processes.
Attendees:	Software project managers and technical contributors who are responsible for complete software engineering efforts.
Description:	Provide the training for software engineers to implement and manage Verification and Validation processes and methods throughout the software life cycle. **Course Outline:** • Foundations for V & V • Overview of V & V Management • Constructing a Life Cycle V & V plan • Concept Phase V & V • Requirements Phase V & V • Design Phase V & V • Implementation Phase V & V • Test Phase V & V • Installation and Checkout Phase V & V • Operation and Maintenance Phase V & V • Software Verification & Validation Reporting • Verification & Validation Administrative Procedures

LR 42

Category of knowledge	Area of knowledge	Unit of knowledge
• Computing Fundamentals (1) • Software Product Engineering (2)	• Programming Languages (1.5) • Software Requirements (2.1) • Software Design (2.2) • Software Coding (2.3) • Software Operation and Maintenance (2.5)	• Programming Paradigms (1.5.2)

Type of Instruction:	Computer Based Instruction
Name of the Institution:	National Program Office for Computer Based Instruction
Name of the Course:	The Software Development Process: Principles
Reference:	http://faawbt.jccbi.gov/cbtlib/htmlplan/cbtweb/curicula/courses/MISG04E/MISG04E.htm
Location:	
Duration:	3 hours
Prerequisites:	Familiarity with a software process
Attendees:	Software process managers, software engineers, team leaders; business managers with responsibility for a software process
Description:	To provide an overview of the software process and principles in world-class software organizations. Topics Covered: Processes and lifecycles – Developing software – Software engineering fundamentals – Software development life cycles – Programming paradigms Software specification and design – Requirements engineering – System design – Formal specification – Computer-aided software engineering – Real-time and distributed systems Reuse and re-engineering – Reuse – Re-engineering

LR 43

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Design (2.2)	

Type of Instruction:	Distance Learning
Name of the Institution:	National Technological University
Name of the Course:	MC99050602 - Software Design Techniques
Reference:	http://www.ntu.edu/5/bytopic99.htm
Location:	One live
Duration:	3-hour broadcast
Prerequisites:	Software development or management familiarity and an understanding of software requirements modeling.
Attendees:	This course is intended for software managers, project leaders, programmers and software engineers who are interested in structured and object-oriented approaches to software development. For those already familiar with basics of software design, this course will serve as a refresher. For those with little or no knowledge of software design techniques, this course will provide a starting point, familiarizing you with concepts, methods and approaches to software design.
Description:	Software design is a mapping from models that represent software requirements to a model for software solutions. Like any other design activity, software design involves judgment and decision making that affects the quality, maintainability, modifiability and other aspects of the end product. A systematic software design approach helps with such decision making and reduces the complexity of this task by a process of stepwise refinement. In this course we will view software design as a translation of the analysis models of the software system into a design layout. The focus is mostly on high-level architectural design aspects of a system, rather than the low-level (algorithmic) design of individual modules/programs. We will introduce basic concepts and principles of software design, and familiarize participants with the most common methods for designing the architecture of software systems. Both function-based and object-based approaches to software design will be presented. BENEFITS: – Understand the issues involved in the process of software design – Identify design objectives and quality criteria for software designs – Apply abstraction techniques and perform stepwise refinement for design – Develop or evaluate software architectures – Perform transform and transaction analysis for dataflow-oriented design – Understand the structure-oriented approach to software design – Understand object-oriented approaches to software design

LR 44

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Requirements (2.1) • Software Design (2.2)	

Type of Instruction:	3 Credit Hour class
Name of the Institution:	National Technological University
Name of the Course:	SE 533-N Software Requirements and Design Engineering
Reference:	http://www.ntu.edu/1/credit/se533n.htm
Location:	Department of Computer Science and Engineering Southern Methodist University P.O. BOX 750122 Dallas, TX 75275 (214) 768-3080 CSE Dept. FAX (214) 768-3085 E-mail: bralick@seas.smu.edu
Duration:	43 (60 minute) lectures
Prerequisites:	None
Attendees:	
Description:	The objective of this course is to impart a solid understanding of the role of requirements engineering and design within the software life-cycle. Students completing this course will have a framework for evaluating structured, object-oriented, data-oriented, and formal approaches to requirements and design and will understand the role of architectural paradigms in the engineering of complex systems. Course Description: The course provides coverage of software requirements engineering with topics that include requirements elicitation, requirements analysis and the development of a software requirements specification. Various approaches to requirements analysis will be examined and a framework for evaluating various approaches will be developed. The coverage of design includes key software design principles and the role of design paradigms and architectures in the design process. Design issues relating to classes of applications including real-time systems and information systems will be explored.

LR 45

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Requirements (2.1)	

Type of Instruction:	Course
Name of the Institution:	Integrated Computer Engineering, Inc., C&CA Division
Name of the Course:	Software Requirements Engineering
Reference:	http://www.iceinc.to/
Location:	2301 Kenstock Drive Suite 103 Virginia Beach, VA 23454-0344 1-888-463-0744
Duration:	1-1/2 days
Prerequisites:	None
Attendees:	In general, anyone involved in software acquisition or development will benefit from this course
Description:	*Software Requirements Engineering* introduces managers, engineers and software specialists to the formal definition, validation and management of software requirements. Requirements Engineering is perhaps the most difficult and critical process in software engineering. All other management and technical activities depend on the complete, accurate and unambiguous definition of system and software requirements. **Objective:** At the end of this course participants should be able to: • Explain why Requirements Engineering is so difficult and so critical • Apply a disciplined process and systematic methods to the identification and specification of requirements • Develop complete, concise, accurate and unambiguous requirements models and specifications

LR 46

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Design (2.2)	

Type of Instruction:	Computer Based Instruction
Name of the Institution:	National Program Office for Computer Based Instruction CBT System Campus Server
Name of the Course:	Object-Oriented Design
Reference:	http://faawbt.jccbi.gov/cbtlib/htmlplan/cbtweb/curicula/courses/OOD/OOD.htm
Location:	
Duration:	3 hours
Prerequisites:	Some experience of conventional systems design
Attendees:	Systems designers
Description:	To provide the user with some knowledge of and experience in using object-oriented methods in the systems design process, with particular reference to database and GUI applications Topics Covered – Refining the object model – Developing objects – Inheritance and delegation – Modifying the object model – Data protection – Documenting the design – Design changes – Validation – Good design criteria – Object-oriented database management – OO/DBMS coupling – Pure OODBMSs – Object store – GUI design – Object window – NextStep

LR 47

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Design (2.2)	

Type of Instruction:	Seminar
Name of the Institution:	Oregon Graduate Institute of Science and Technology Center for Professional Development
Name of the Course:	Object Oriented Analysis and Design
Reference:	http://www.ogi.edu/CPD/courses/
Location:	20000 N.W. Walker Road Beaverton, Oregon 97006-8921 Fax: +1-503-748-1686.
Duration:	4 days
Prerequisites:	Students are expected to have a background in programming and/or software systems design.
Attendees:	
Description:	At the end of the seminar, participants will be able to describe the concepts and expected benefits of object oriented programming, recognize the characteristics of a design that will actually get those benefits, understand design patterns and describe at least three of them, understand frameworks and the structure of object oriented libraries, read UML notation and use it to develop and communicate a design, understand use cases and apply use case analysis and design read object interface descriptions in C++, Java and CORBA IDL, understand the Java and CORBA models for concurrent objects, compare and contrast the CORBA and OLE distributed object models, plan a development project using an interative/incremental approach.

LR 48

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Requirements (2.1)	

Type of Instruction:	Computer Based Instruction
Name of the Institution:	National Program Office for Computer Based Instruction CBT System Campus Server
Name of the Course:	Object-Oriented Analysis - Objects and Classes
Reference:	http://faawbt.jccbi.gov/cbtlib/htmlplan/cbtweb/curicula/courses/OOAOC/OOAOC.htm
Location:	
Duration:	5 hours
Prerequisites:	Some experience of conventional systems analysis
Attendees:	Systems analysts
Description:	To provide the user with some knowledge of and experience in using object-oriented methods in the systems analysis process' Topics Covered – Software development models – Objects – Functions vs objects – Challenging candidate objects – Object descriptions – Class hierarchies – Class responsibilities – Methods – Class attributes – Object diagrams – Messages – Class relationships – Generalization and specialization – Inheritance – Aggregate relationships – Association – Object schemas

LR 49

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Requirements (2.1)	

Type of Instruction:	Seminar
Name of the Institution:	Oregon Graduate Institute of Science and Technology Center for Professional Development
Name of the Course:	Software Requirements Engineering
Reference:	http://www.ogi.edu/CPD/courses/
Location:	20000 N.W. Walker Road Beaverton, Oregon 97006-8921 Fax: +1-503-748-1686.
Duration:	2 days
Prerequisites:	No special background is required. Those with more experience and knowledge of software engineering issues will benefit most from this course.
Attendees:	Product managers, software project managers, software acquirers, software developers, systems engineers, line managers and all others who prepare requirements, approve them, or live with the consequences of requirements analysis will benefit from this course.
Description:	This two-day course is one in a series of courses offered by the Center for Professional Development of Oregon Graduate Institute in collaboration with the Competency Recognition Program of the IEEE Computer Society. The course is based on IEEE Standards 830 and 1362. Deficiencies in requirements is one of the primary reasons software projects fail to deliver satisfactory products within acceptable time frames and resource allocations. Software requirements engineering involves understanding user needs and customer expectations and mapping those needs and expectations into technical specifications for a system that will satisfy users and customers in a complete, consistent, and unambiguous manner. Requirements engineering is also concerned with managing the evolving baseline and conducting impact analyses as requirements change. The course covers methods, tools, and techniques for eliciting, analyzing, documenting, verifying, and managing requirements. Outline: 1. Introduction to requirements engineering 2. User requirements and technical specifications 3. Elicitation and analysis of requirements 4. Prioritizing and verifying the requirements 5. Mapping user requirements into technical specifications 6. Techniques for documenting software requirements 7. Managing the evolution of software requirements

LR 50

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Design (2.2)	• Architectural Design (2.2.1)

Type of Instruction:	Seminar
Name of the Institution:	Advanced Strategies, Inc.
Name of the Course:	Architectural Design
Reference:	http://www.advancedstrategiesinc.com/courses/c37.htm
Location:	3980 Dekalb Technology Parkway in Atlanta, Ga.
Duration:	Two Days
Prerequisites:	Data Modeling and Process Modeling
Attendees:	This course is targeted for analysts, designers, and other individuals involved in system design.

Description:	This course presents the concepts of architectural design and illustrates how to document the architectural design using the new physical data flow diagram. It uses three models created during analysis as a basis for developing the new physical data flow diagram: The Data Model, The Process Model, and The Event Model.
	The student shall gain the ability to employ a reliable, orderly, and manageable means for using the analysis outputs to develop a system architectural design.
	
	Course Outline:
	– The Analysis Models
	– Reconciling Analysis Models
	– CRUD and Other Correlations
	– Data Transaction Analysis
	– Developing the New Physical Data Flow Diagram
	  – Business System Design
	  – Technical Design

LR 51

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Design (2.2)	

Type of Instruction:	Seminar
Name of the Institution:	Advanced Strategies, Inc.
Name of the Course:	Object-Oriented Analysis and Design
Reference:	http://www.advancedstrategiesinc.com/courses/c38.htm
Location:	http://www.advancedstrategiesinc.com/courses/c12.htm
Duration:	Four Days
Prerequisites:	Event Modeling and Process Modeling
Attendees:	Individuals who are charged with designing business applications which are to be implemented with an object-oriented programming language.
Description:	This course provides a way for software developers to successfully utilize the new object-oriented software programming languages by shifting their development paradigm and engineering their applications appropriately. This course will cover a strategy for transforming business events into system events and mapping those into GUI components. The student shall gain the skills necessary to transform the analysis specification into an engineering design of an event-driven graphical user interface (GUI) application that uses software objects. The student will also learn the fundamentals of designing GUI interfaces and appropriate uses for them The course will also provide the student with concepts and a strategy for isolating object-oriented portions of applications from traditional portions. Course Outline: – Analysis specifications needed to support object-oriented design – Graphical User Interface (GUI) principles, components, and uses – Analysis of high-level business events and detail events – Object-oriented programming concepts – Using objects vs. creating objects – Encapsulation

LR 52

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Requirements (2.1)	

Type of Instruction:	Seminar
Name of the Institution:	Software Productivity Consortium
Name of the Course:	Requirements Management
Reference:	http://www.software.org/pub/Courses.html
Location:	SPC Building 2214 Rock Hill Road Herndon, VA 20170-4227 1-703-742-7211
Duration:	3 days
Prerequisites:	
Attendees:	This course is for project managers, software managers, senior software staff and software engineers responsible for the definition, allocation, or management of requirements.
Description:	Managers and developers cite continuous requirements changes as being among the most significant problems they face. The Requirements Management course addresses the management controls that must be established and followed before user requirements can be correctly captured and consistently implemented. This course will offer methods to reduce the impact of requirements changes by capturing them correctly the first time, creating baselines, and managing changes to requirements as they evolve. By controlling the requirements process, participants learn to schedule and cost software activities more accurately, and maintain consistency among software plans, products and activities. **Benefits** – Upon completion of this course, attendees will be able to: – Identify CMM® Level 2 Requirements Management activities – Establish a baseline for software requirements – Define a process for reviewing requirements – Evaluate requirements for consistency and correctness – Identify techniques for tracing requirements through the software life cycle – Define a process for managing changes to requirements.

LR 53

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Requirements (2.1)	

Type of Instruction:	Seminar
Name of the Institution:	Software Productivity Center
Name of the Course:	In Search of Excellent Requirements
Reference:	http://www.spc.ca/training/courses/990506-req.htm
Location:	#460-1122 Mainland St., Vancouver, BC V6B 5L1, Canada Tel: (604) 662-8181 Fax: (604) 689-0141
Duration:	2 days
Prerequisites:	none
Attendees:	This seminar will be useful for software engineers, managers, requirements analysts, and anyone else engaged in gathering, documenting, analyzing, or managing customer requirements for software applications.
Description:	This seminar describes tested methods that can help any organization improve the way it elicits, analyzes, documents, verifies and manages software requirements. Characteristics of excellent requirements statement and requirements specifications are presented and used to evaluate some sample functional requirements. The seminar emphasizes several practical techniques: – customer involvement through a "project champion" model – the application of use cases for defining user needs and system functions – a simple model for prioritizing requirements writing software requirements specifications using a standard template – construction of dialog maps to model user interfaces – the use of prototypes to clarify and refine user needs – the use of technical inspections to find requirements errors – use of a requirements traceability matrix to connect requirements to design elements, code, and tests. The basic concepts of requirements management are described, as are practical methods for managing changes to requirements. These techniques can reduce project risk by improving the quality and control of the software requirements, thereby increasing the likelihood of a successfully completed project.

LR 54

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Requirements (2.1)	

Type of Instruction:	Seminar
Name of the Institution:	GTE – Information Technology Training
Name of the Course:	Requirements Management Using RequisitePro
Reference:	http://ittraining-gte.com/itt/default2.asp?mainURL=Catalog/default.asp
Location:	Tampa
Duration:	2 Days
Prerequisites:	– Requirements Management Fundamentals class. – Mandatory Reading: Software Requirements Engineering (Chapter 3). – Students must bring their copy of Software – Requirements Engineering to class (provided in Module 1). – Optional Reading: RAPID Development by Steve McConnell. – Must bring an example of a recent requirements – document that they are in the process of creating. – As an option, students can bring a completed project. – Material must be in soft copy on a standard 3.5” – 1.44Mb High Density Floppy Disk in MSWord format. – This material is critical for the second day of this course.
Attendees:	This course is intended for all individuals involved in the activity of gathering, documenting and managing requirements allocated to software.
Description:	Using both lectures and exercises, students will learn the functionality of the RequisitePro case tool its use in performing Requirements Management. Emphasis is provided in areas of proper requirements documentation and traceability. OBJECTIVES: – Use the RequisitePro software package (case tool). – Understand how RequisitePro is used to document – and manage requirements allocated to software. – Tag and relate requirements documented in the SRS. – Establish and manage requirement document – hierarchies. – Use automated requirements traceability features. – Convert a real SRS into RequisitePro.

LR 55

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Requirements (2.1)	

Type of Instruction:	Seminar
Name of the Institution:	Software Productivity Consortium
Name of the Course:	Consortium Requirements Engineering (CoRE)
Reference:	http://www.software.org/pub/Courses.html
Location:	SPC Building 2214 Rock Hill Road Herndon, VA 20170-4227 1-703-742-7211
Duration:	
Prerequisites:	3-day video, ADARTSSM /CoRE 1/2-day overview
Attendees:	Software engineers who develop requirements for real-time embedded systems. The participants should have basic skills in embedded system development and a general knowledge of modern software engineering principles and practices.
Description:	This 3-day course provides instruction in applying the CoRE method. Benefits of CoRE include the ability to develop precise, testable requirements specifications, which are demonstrably complete and consistent, for embedded software systems. The course covers CoRE benefits, foundational concepts, the CoRE process, notation, and the analysis of CoRE specifications for completeness and consistency. The class exercises provide experience in the method through hands-on application. On completion of this course, attendees will understand: – The goals and benefits of the CoRE method – The underlying concepts and principles used to develop rigorous requirements specifications in CoRE – Notations and specification techniques for writing a CoRE specification – How to develop a requirements specification to facilitate requirements changes, risk – mitigation, and reuse – How to analyze a CoRE specification for completeness and consistency

LR 56

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Requirements (2.1)	

Type of Instruction:	Workshop
Name of the Institution:	STSC
Name of the Course:	Requirements Engineering
Reference:	http://www.stsc.hill.af.mil/pns/requirements/rewkshp.asp (801)775-3055 DSN 775-3055 Email: cookd@software.hill.af.mil
Location:	RE Workshop is conducted at client sites.
Duration:	4 half days (with afternoon mentoring) or per client request.
Prerequisites:	
Attendees:	Organizations that are: Exploring the scope of their requirements issues, Wishing to learn about "Engineering" requirements; Implementing a requirements process change.
Description:	Three themes are developed during the workshop 1.The benefits of "Engineering" Requirements. 2.Technology Adoption. 3.Process Improvement. – The Definition of Requirement – Requirements and the Life Cycle – Requirements Management – Requirements Elicitation – Requirements Analysis – Object Oriented Analysis – Structured Analysis – Real Time Analysis – Managment indicators for Quality Requirements – Requirements Tools and Techniques – Requirements Documentation – Requirements V&V – Quality Requirements – Writing testable Requirements – Technology Adoption – Process Improvement

LR 57

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Requirements (2.1)	

Type of Instruction:	Seminar
Name of the Institution:	Richard Ball & Associates Inc.
Name of the Course:	Software Requirements Gathering & Specification
Reference:	http://www3.pei.sympatico.ca/ball/index.html
Location:	
Duration:	2 days
Prerequisites:	None
Attendees:	– Software Requirements Analysts – Software Developers – Business Analysts – Business Users involved in the software requirements process – QA and Audit – Software Process Engineers
Description:	This course provides a comprehensive introduction to state-of-the-art software requirements gathering and specification techniques. It defines where requirements fit into the SDLC. It presents current industry standards for software requirements gathering and specification - including the IEEE and Software Engineering Institute (SEI) requirements standards and guidelines. It clearly contrasts requirements (problem) statements from design (solution) statements. Course Outline: – Roles & Responsibilities – "Problem" vs. "Solution" – The SW Requirements Definition Process – Requirements Building Blocks – The Software Requirements Specification (SRS) – Automated Tools for Requirements Gathering and Specification – Requirements Diagramming, Modeling, and Methodologies – Requirements Gathering Techniques (I) – Requirements Gathering Techniques (II) – Requirements Validation Techniques – Requirements Methodology Framework

LR 58

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Testing (2.4)	

Type of Instruction:	Seminar
Name of the Institution:	Rice Consulting Services, Inc.
Name of the Course:	Becoming an Effective Test Team Leader
Reference:	http://www.riceconsulting.com/
Location:	
Duration:	2 days
Prerequisites:	A basic knowledge of software testing techniques.
Attendees:	This session is designed for test leaders and test managers, people who expect to be in a test leadership role, or people who lead other test managers and test leaders. Any developer, tester, user or manager who wants to learn about the basics of software testing. Any tester, user or manager who wants to learn about what it takes to successfully and effectively lead a software testing effort.
Description:	The main objective of this session is to teach you how to be the very best test manager and leader. We will discuss what makes a good leader and how to be the best at leading a test team. How can we keep up with changes and still test to deliver quality software? This session also seeks to answer that question. A good test team leader must also know the basic issues involved in testing. To reinforce this knowledge, this session will present an overview of: – Regression testing and why you need to do it – Regression testing issues and tips for performing regression testing – Test automation vs. manual testing – Tips for automated testing – Test planning and tips for test planning – How to recruit and train a quality test team – Working with developers and users – How to keep the test on track After attending this session, you will have the information to effectively lead a test team, make your case to management, and lead your test team through the challenges you face daily.

LR 59

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Testing (2.4)	• Integration Testing (2.4.2) • System Testing (2.4.3)

Type of Instruction:	Distance Learning
Name of the Institution:	Northern Virginia Community College (NVCC)
Name of the Course:	0602-835 - Software Testing and Implementation
Reference:	http://distancelearning.rit.edu/MsinsoftwareDevelpmentMgr.html
Location:	RIT Distance Learning Local #:716-475-5089 Phone: #1-800-CALL RIT TTY #: 716-475-5896 Fax #: 716-475-5077 Email: DISTED@rit.edu
Duration:	Lecture 3 hours per week.
Prerequisites:	Part of a Masters in Software Development and Management: A certain minimal background is required of all students wishing to enter the master's program. Acceptance into the master's program is possible even though the applicant must accomplish some additional courses. Students whose undergraduate or industrial preparation does not satisfy the above content or grade-point requirements may satisfy requirements by taking one or more of the following Bridge Program courses, as prescribed by the Graduate Program Chair. Courses in the Bridge Program are not part of the 48 quarter credit hours required for the master's degree, and their grades are not included in the student's graduate grade-point average. Courses in the student's academic background and work experience can be used to satisfy these prerequisites, with approval of the department. Remaining prerequisites must be met by completing appropriate courses at a post-secondary institution of the student's choosing. Courses selected for this purpose must be approved by the department.
Attendees:	Any
Description:	Topics covered include testing schemes (black-box, white-box), integration schemes, validation testing, graphic analysis. Reliability models (seeding, hazard) are covered. Software maintenance techniques and tools are covered. (0602-820) Credit 4

LR 60

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Testing (2.4)	• Test Documentation (2.4.7) • Unit Testing (2.4.1)

Type of Instruction:	Seminar
Name of the Institution:	Computer Generated Solutions - Instructor Led Technical Training
Name of the Course:	Software Quality Assurance Techniques
Reference:	http://devel.netxactics.com/cgsinc/_training/ilt/t272.htm
Location:	Computer Generated Solutions, Inc. (World Headquarters) 1675 Broadway, New York, N.Y. 10019 Tel: (212) 408-3800 • Fax: (212) 977-7474
Duration:	2 Days
Prerequisites:	None
Attendees:	Programmers, analysts, business liaisons and project leaders
Description:	This course offers techniques to locate errors early in development and also concentrates on the more labor-intensive and dynamic coding and testing phases. Frequently, testing is the most expensive, but also ineffective phase of a project. The purpose of testing is not to create specifications, but to ensure that the system being developed meets the agreed-upon specifications. Quality is a major goal for customers and systems. Although testing does not create quality, it certainly helps us determine if we have it. This course provides the tools to save time, money, and frustration, and deliver systems that are more effective. Why testing, when and who – Definitions, terms and categories – The test plans – The data – Dynamic testing techniques – Testing design – Testing analysis – Testing implementation

LR 61

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Testing (2.4)	

Type of Instruction:	Seminar
Name of the Institution:	Richard Ball & Associates Inc.
Name of the Course:	Software Testing and Quality Assurance
Reference:	http://www3.pei.sympatico.ca/ball/index.html
Location:	
Duration:	3 days
Prerequisites:	None
Attendees:	– IT practicing professionals and Quality Assurance staff
Description:	Testing and quality assurance skills are increasingly essential to successful software development. System Requirements and Design must be tested for completeness, consistency, and clarity. Programs must be tested for correctness, structure, simplicity, and efficiency. Delivered software must be tested for function, performance, usability, reliability, etc. and then integrated into an increasingly complex production environment. This seminar presents state-of-the-art software testing and quality assurance techniques covering the entire software lifecycle. The seminar maintains a practical perspective, dealing realistically with issues such as deadline-driven and budget-constrained testing efforts. **Course Outline:** Software Testing: The Problem Lifecycle Testing: The Solution Initiation-Phase Testing Requirements-Phase Testing Design-Phase Testing Testing During SW Build & Implementation The Software Testing Lifecycle (STLC) Requirements-Based (Black Box) Test Case Design Code-Based (White Box) Test Case Design Unit Testing System, Acceptance, and Production-level Testing Build/Thread Testing: An Integrated Development/Test Strategy

LR 62

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Testing (2.4)	

Type of Instruction:	Seminar
Name of the Institution:	Forum Training
Name of the Course:	Software Testing: Part I
Reference:	http://www.swforum.com/training/testing1.html
Location:	
Duration:	1.5 Days
Prerequisites:	None
Attendees:	Software developers and testers who perform unit and integration testing in software environments
Description:	This course presents a foundation for planning and executing efficient and effective unit and integration testing. Participants learn the importance and benefits of testing and how to apply systematic functional coverage and code coverage techniques. They also learn how to perform incremental integration and regression testing and how to use test process assessment and improvement techniques. Planning an actual testing activity is part of the course. Upon completion, the participant will be able to: • Understand the importance and benefits of testing • Plan a testing activity • Apply systematic functional coverage techniques • Apply systematic code coverage techniques • Perform incremental integration and regression testing • Utilize test process assessment and improvement techniques

LR 63

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Testing (2.4)	

Type of Instruction:	Distance Learning
Name of the Institution:	National Technological University
Name of the Course:	MC99081801 Statistical Testing for Software Intensive Systems
Reference:	http://www.ntu.edu/5/bytopic99.htm
Location:	One live
Duration:	3-hour broadcast
Prerequisites:	A basic knowledge of software testing practices and statistics.
Attendees:	This course will be of interest to technical software developers, managers, software quality engineers, and systems engineers who deal with the many issues confronting the testing software and software intensive systems.
Description:	In many industries, products are typically certified using protocols in which random samples of the products are drawn, test characteristics of operational use are applied, and analytical or statistical inferences are made. Products meeting this pre-defined standard are "certified" as fit for use. The same concepts can be applied to software. Using the same statistical base, software can be tested using operational usage models where test cases are generated randomly from the usage model and the test results are interpreted according to mathematical and statistical models to determine measures of software reliability. In this course, the student will be introduced to the techniques used to test software using operational usage models. The student will be introduced to the steps required to generate these models for software testing including test planning, user and usage stratification, model generation, test generation, and test result analysis.

LR 64

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Operation & Maintenance (2.5)	

Type of Instruction:	Workshop
Name of the Institution:	Esprit Systems Consulting, Inc.
Name of the Course:	Reverse Engineering & Software Maintenance
Reference:	http://www.espritinc.com/pages/structuredseminar/strucsem.html
Location:	P.O. Box 1486, West Chester, PA 19380 Phone: (610) 436-8290 Fax: (610) 436-9848 Email: esprit@espritinc.com
Duration:	Three Day
Prerequisites:	Coding Experience
Attendees:	This course is intended for software engineers, maintenance engineers, test engineers, or anyone responsible for the quality and management of existing software systems.
Description:	A coherent methodology for using Yourdon/Constantine structure charts to preserve, understand and improve legacy software is the subject of these three days. Participants discuss designs, reverse engineer code and evaluate approaches to legacy Management.

LR 65

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Operation & Maintenance (2.5)	

Type of Instruction:	Workshop
Name of the Institution:	ICSM '99
Name of the Course:	INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE - 1999
Reference:	http://www.cms.dmu.ac.uk/ICSM99/ap.html#tutorials
Location:	Keble College, Oxford, England
Duration:	30 August - 3 September, 1999
Prerequisites:	
Attendees:	
Description:	Tutorial T1 (Monday, 30 August, 09.00-17.30): **Understanding and applying the Unified Modeling Language (UML)** The Unified Modeling Language (UML) is an Object Management Group standard object modeling notation. This tutorial develops an understanding of UML models that can lead to effective use. Tutorial T2 (Monday, 30 August, 09.00-12.30): **Designing and evaluating program understanding tools** This tutorial reviews techniques and tools to assist maintainers in program comprehension and suggests methods for designing or selecting a suitable tool for a particular project. Topics include cognitive models of program comprehension, information needs during program understanding, and the evaluation of program understanding tools. It helps to answer the following three questions: Tutorial T3 (Monday, 30 August, 09.00-17.30): **Automated software evolution: From problem to solution** This tutorial illustrates establishing an architecture for automated software evolution in order to automate maintenance and reengineering tasks, with live demonstrations of the tools that are discussed. Tutorial T4 (Monday, 30 August, 09.00-12.30): **Maintenance from the reuse perspective: A synthesis of industrial experiences** This tutorial introduces the essential concepts of software reuse, how they relate to maintenance, and how to introduce reuse in companies producing software or systems. A large part of the tutorial is examples and lessons learnt from industrial projects. The tutorial is the result of the Esprit Project SURPRISE (SURvey on the Possibilities of Reuse In Software Engineering).

Tutorial T5 (Monday, 30 August, 09.00-17.30):
Software maintenance cost estimation
- Special considerations of maintenance cost estimation
- Maintenance requirements analysis
- Methods of maintenance cost estimation
- Measuring the effect of the existing system
- Making the estimate
- Calibrating the estimate
- Demonstration of automated impact analysis
- Demonstration of automated cost estimation

Tutorial T6 (Monday, 30 August, 14.00-17.30):
Object-oriented re-architecting
The tutorial presents our tool-supported object-oriented re-architecting method CORET for re-structuring and transforming applications from C to C++. It presents the principles and methods of object-oriented re-architecting, how to manage uncertainties, how to integrate the human into the process, and a representative case study in C.

Tutorial T7 (Monday, 30 August, 14.00-17.30):
Software surgery
Pat the Programmer, a developer or maintainer, stares at a piece of code and contemplates a change. Pat asks the following:
1. Can I analytically determine if the change is as small as I believe?
2. Can I be sure that I don't introduce new errors with the change?
3. Are there any approaches to changing the code that will minimize my effort?

The answer to Pat's questions are "Yes!" This tutorial will show software engineers how to put boundaries on the effects of a
change and guarantee that no new errors are introduced. The method is applicable to a wide variety of programming languages.

Tutorial T11 (Tuesday, 31 August, 09.00-17.30):
Round-trip engineering with design patterns, UML, Java and C++
The tutorial presents the state-of-the-art in methodologies and tools for round-trip engineering of object-oriented software systems. We present: first, semi-automatic derivation of implementations from design documents; second, the translation of UML behavior diagrams to Java or C++ code; third, design patterns providing additional information and means for the derivation of a valid implementation.

Tutorial T12 (Tuesday, 31 August, 09.00-17.30):
A primer on empirical studies
This tutorial provides a sound empirical basis for software and process engineering and research by focusing on empirical studies. The primary goal is to enable the attendees to assess the credibility of empirical work either as reported in the software engineering literature or as done by themselves and to apply the results to their own work. We note that good empirical science is the result of iterative experimentation and we use this basis to establish criteria for evaluating both the experimental structures and the experimental results. We show how to exploit data. We discuss the position of statistics in our model and the importance of minimal manipulation of data. We present new techniques such as simulation and sampling.

Tutorial T13 (Tuesday, 31 August, 09.00-17.30):
MORALE: Architectural support for evolution of legacy systems
This tutorial presents an architecture-centric approach to the evolution of legacy software systems. The tutorial revolves around a

	selection of case studies undertaken by the MORALE project, funded by DARPA, a suite of methods and tools to support the understanding of the implementation and architecture of complex systems as they undergo mission-oriented evolution. Tutorial T14 (Tuesday, 31 August, 09.00-17.30): **Measuring and evaluating the development and maintenance process using reliability, risk, test, and complexity metrics** This tutorial examines the relationship between product quality and process stability. In analyzing the stability of a development and maintenance process, it is important that it not be treated in isolation from the reliability and risk of deploying the software that result from applying the process. An extensive collection of reliability, test, effort, and metrics data from the NASA Space Shuttle is used as an example application of the unified approach. Tutorial T15 (Tuesday, 31 August, 09.00-17.30): **Software life cycle management** The tutorial shows how to predict and meet the maintenance demands of a much-used system, from immediate requests for operational support and unpredictable episodes of trouble-shooting and correction, to longer term needs for responsive changes and to ultimate replacement. It specifies the key methods of control used to maintain reliable service under the flux of change, how to organize for immediate support and problem-solving, and how to design and install functional changes.

LR 66

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Operation & Maintenance (2.5)	

Type of Instruction:	Self tutorial
Name of the Institution:	Department of Computer Science University of Victoria
Name of the Course:	Reverse Engineering Tutorial, Understanding Software Systems Using Reverse Engineering Technologies Research and Practice
Reference:	http://www.rigi.csc.uvic.ca/UVicRevTut/UVicRevTut.html
Location:	
Duration:	
Prerequisites:	
Attendees:	
Description:	– Software evolution – Definitions and context – Reengineering strategies – Reverse engineering – Program understanding – Selected reverse engineering tools – Selected research projects – Selected research challenges

LR 67

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Operation & Maintenance (2.5)	

Type of Instruction:	Self Tutorial
Name of the Institution:	SEI
Name of the Course:	Legacy System Reengineering
Reference:	http://www.sei.cmu.edu/reengineering/pubs/lsysree/lsysree.html
Location:	
Duration:	
Prerequisites:	
Attendees:	
Description:	1. An engineering perspective 2. A system perspective 3. A software perspective 4. A managerial perspective 5. An evolutionary perspective 6. A maintenance perspective

LR 68

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Product Engineering (2)	• Software Operation & Maintenance (2.5)	

Type of Instruction:	Seminar
Name of the Institution:	Richard Ball & Associates Inc.
Name of the Course:	Software Maintenance Strategies and Techniques
Reference:	http://www3.pei.sympatico.ca/ball/index.html
Location:	
Duration:	3 days
Prerequisites:	None
Attendees:	This seminar is aimed at staff involved in performing software maintenance activities: – Software maintenance practitioners, team leaders, and supervisors – User coordinators, contacts, and business analysts – Software developers, Q/A staff and IT auditors
Description:	RB&A has the most extensive software maintenance curriculum available. While working for Imperial Oil, Richard Ball developed a software maintenance methodology that was adopted as an international standard within Exxon Corporation. The comprehensive nature of this methodology is reflected in RB&A's software maintenance training. Course Outline: – Software Maintenance: The Problem – Software Management: The Solution – Roles & Responsibilities – Software Acceptance Process: The Key to Delivering Maintainable Software – Problem Management Process: Solving Operational Difficulties – Service Request Process: Key to Controlling Software Change – Scheduled Release (SR) Process: Key to Improved Productivity & Quality – Release Development Process – Build/Thread Testing: An Integrated Release Development/Test Approach – System Performance Monitoring – Software Improvement Process: Key to Continuous Improvement – Software Maintainability/Usability Evaluation Process – Documentation Strategies – Motivating and Managing SWM Staff

LR 69

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Acquisition (3.6)	• Procurement Management (3.6.1)

Type of Instruction:	University – Semester course
Name of the Institution:	Northern Virginia Community College (NVCC)
Name of the Course:	ACQ 121 INTRODUCTION TO ACQUISITION AND PROCUREMENT FUNDAMENTALS I
Reference:	http://www.nv.cc.va.us/catalog/cat98/descript/acq121.htm
Location:	4001 Wakefield Chapel Road Annandale, Virginia 22003-3796 Telephone: (703) 323-3000
Duration:	Lecture 3 hours per week.
Prerequisites:	None
Attendees:	Any
Description:	Introduces technical and fundamental procedures of government acquisition and procurement. Focuses on appropriations and funding, competition requirements, types of specifications, and contractor qualifications. (For those institutions certified, satisfies requirements of the mandatory Department of Defense (DOD) course, Contracting Fundamentals, when combined with ACQ 122 and DOD materials.)

LR 70

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Acquisition (3.6)	

Type of Instruction:	Seminar
Name of the Institution:	National Capital Training Center
Name of the Course:	
Reference:	http://grad.usda.gov/reg/cat9943b2.html#On-site 600 Maryland Avenue SW, Suite 280 Washington, DC 20024-2520 Phone: (202) 314-3400 FAX: (202) 479-6810 TDD: (202) 314-3450 TOLL-FREE: (888) 744-GRAD
Location:	On site
Duration:	2 to 5 days
Prerequisites:	
Attendees:	
Description:	The following courses are available for delivery on-site at a location of your choice. For more information, contact the Training Center serving your state or call (202) 314-3406. Acquisition Management for Executives and Managers 43TC 2 Days FaxBack # 1720 Basic Lease Contracting 43TD 5 Days FaxBack # 1731 Construction Contracting 43TE 5 Days FaxBack # 1738 Federal Contracts: Default and Procurement 43PX 5 Days FaxBack # 1760 Formation of Government Contracts 43TF 3 Days FaxBack # 1878

LR 71

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Acquisition (3.6)	

Type of Instruction:	Seminar
Name of the Institution:	National Capital Training Center
Name of the Course:	Acquisition Reform Overview
Reference:	http://grad.usda.gov/reg/cat9943b1.html#43PL
Location:	600 Maryland Avenue SW, Suite 280 Washington, DC 20024-2520 Phone: (202) 314-3400 FAX: (202) 479-6810 TDD: (202) 314-3450 TOLL-FREE: (888) 744-GRAD
Duration:	1 day
Prerequisites:	
Attendees:	Federal, state and local government personnel and others involved in the federal contracting process.
Description:	Become familiar with the major changes resultant from the new procurement reform legislation ; Gain an understanding of federal acquisition laws: – Federal Acquisition Streamlining Act (FASA) – Techniques for micro-purchases – The Simplified Acquisition Methodology – The Federal Acquisition Computer Network (FACNET) – Procurement of commercial items – Changes to the Truth in Negotiations Act (TINA) – Changes in task and delivery order contracting methods – Federal Acquisition Reform Act (FARA) – Streamlined competition requirements – Simplified procedures for commercial item acquisitions up to $5 million – Changes in debriefing procedures – Decoupling FACNET from the use of simplified acquisition procedures – Revision of procurement integrity laws – Information Technology Management Reform Act (ITMRA) – Repeal of central authority of the GSA Administrator – Repeal of the authority of General Services Board of Contract Appeals to hear IT protests – Revision of the process for acquisitions of Information Technology – Expansion of protest authority of the Comptroller General

LR 72

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Acquisition (3.6)	• Procurement Management (3.6.1) • Acquisition Planning (3.6.2)

Type of Instruction:	Computer Based Instruction
Name of the Institution:	National Program Office for Computer Based Instruction
Name of the Course:	Project Management: Procurement
Reference:	http://faawbt.jccbi.gov/cbtlib/htmlplan/cbtweb/curicula/courses/PROJ12E/PROJ12E.htm
Location:	
Duration:	4 hours
Prerequisites:	A general understanding of the nature of projects; completion of the course Project Management; Fundamentals
Attendees:	Trainee project managers; consultants in project management; experienced project managers who need a refresher of evolving technologies; senior managers who employ or manage project managers
Description:	To examine the procedures whereby goods and services may have to be procured from outside sources for the project, including an examination of the various types of contract that may be required Topics Covered Procurement planning – Assessing procurement needs – Types of contract – Procurement management plan Solicitation – Solicitation planning – The solicitation process – Awarding the contract Contract administration and closeout – Administering the contract – Contract change control – Contract closeout

LR 73

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Acquisition (3.6)	• Procurement Management (3.6.1)

Type of Instruction:	Course
Name of the Institution:	Federal Aviation Administration Center for Management Development
Name of the Course:	05603, Acquire Purchasing Training
Reference:	http://www.academy.jccbi.gov/
Location:	FAA ACADEMY & CMD TRAINING COURSES
Duration:	24 hours
Prerequisites:	None
Attendees:	FAA procurement specialists, contracting officers, and others
Description:	The objective of the acquire training is to train FAA procurement specialists, contracting officers, and others on the new Acquire system being installed to replace the current System for Acquisition Management (SAM). Training will be lecture and hands-on activities. Training will be structured around requisitioning, requisition approval, and fund certification activities. The four components are; 1) functional training; 2) business process training; 3) functional overview; 4) purchasing overview. Each trainee will receive an Acquire training manual to achieve course objectives and to use as a reference guide for the system and an instructional guide.

LR 74

Category of knowledge	Area of knowledge	Unit of knowledge
• Software Management (3)	• Software Acquisition (3.6)	• Procurement Management (3.6.1)

Type of Instruction:	Seminar
Name of the Institution:	Office of Acquisition Management Training and Certification
Name of the Course:	See catalog
Reference:	http://www.os.dhhs.gov/progorg/oam/training.html
Location:	See catalog
Duration:	See catalog
Prerequisites:	
Attendees:	
Description:	The Procurement Training Program is designed to develop a highly qualified professional procurement workforce. The Procurement Training Program consists of fourteen core and seven specialized courses for individuals in procurement positions. OAM provides leadership in the development of procurement policy and conducts performance measurement of the HHS acquisition program; fosters procurement innovation and reform; manages the acquisition training and certification program; defends bid protests; and maintains the HHS procurement data reporting system. OAM serves as liaison to Congress, OMB, GAO, and other Federal agencies.